STRATEGY TO MINIMIZE INFORMATION TECHNOLOGY INCIDENTS AND THEIR IMPACT AT PT INOVINDO NUSAKARYA, A FINTECH COMPANY

FINAL PROJECT

In partial fulfilment of the requirements
for the master's degree
from Institut Teknologi Bandung

By
FARIS ARIFIANSYAH
Student ID: 29123055
(Master of Business Administration Program)

INSTITUT TEKNOLOGI BANDUNG
April 2025

ABSTRACT

STRATEGY TO MINIMIZE INFORMATION TECHNOLOGY INCIDENTS AND THEIR IMPACT AT PT INOVINDO NUSAKARYA, A FINTECH COMPANY

By
Faris Arifiansyah
Student ID: 29123055
(Master of Business Administration Program)

The reliability of Information Technology (IT) systems is crucial for technology-driven businesses, as system failures can cause financial losses, operational disruptions, and customer dissatisfaction. Despite having an incident management framework, PT INUSA has experienced recurring IT incidents, indicating inefficiencies in incident prevention, detection, and response. This study investigates the root causes of major IT incidents and evaluates gaps in detection and resolution to identify areas for improvement.

This research adopts a qualitative approach by conducting thematic analysis on 26 post-mortem reports from August 2023 to August 2024. The findings reveal that 80% of incidents were triggered by internal changes, highlighting weaknesses in validation, testing, and deployment controls. Systemic issues were identified, including deficiencies in testing coverage, deployment and change control, and configuration handling. Additionally, the study found that 69% of major incidents lacked proper alerts, leading to delayed detection and prolonged Mean Time to Detect (MTTD) of 82.5 hours. Response inefficiencies, such as delayed escalation and inadequate post-fix monitoring, contributed to a Mean Time to Resolve (MTTR) of 38.5 hours, significantly exceeding industry benchmarks.

To address these challenges, this research proposes structured improvements in IT incident management. Key recommendations include enhancing automated testing, improving deployment and change review procedures, enforcing structured on-call escalation procedures, and standardizing alerting mechanisms. Additionally, post-fix monitoring practices should be strengthened to ensure the effectiveness of issue resolution.

This study contributes to IT incident management by providing data-driven insights into systemic failures and practical strategies for reducing incident occurrence, improving monitoring coverage, and optimizing response workflows. By implementing these improvements, PT INUSA and similar fintech organizations can enhance system resilience, reduce operational risks, and minimize business impact from IT failures.

Keywords: IT Incident Management, Incident Prevention, Incident Detection, Fintech

ABSTRAK

STRATEGI UNTUK MEMINIMALKAN INSIDEN TEKNOLOGI INFORMASI DAN DAMPAKNYA DI PT INOVINDO NUSAKARYA, SEBUAH PERUSAHAAN FINTECH

Oleh
Faris Arifiansyah
29123055
(Program Studi Magister Administrasi Bisnis)

Keandalan sistem Teknologi Informasi (TI) sangat penting bagi bisnis berbasis teknologi, karena kegagalan sistem dapat menyebabkan kerugian finansial, gangguan operasional, dan ketidakpuasan pelanggan. Meskipun PT INUSA telah memiliki kerangka kerja manajemen insiden, perusahaan masih mengalami insiden TI berulang, yang mengindikasikan adanya ketidakefisienan dalam pencegahan, deteksi, dan respons insiden. Penelitian ini menyelidiki akar penyebab insiden besar serta mengevaluasi kekurangan dalam deteksi dan penyelesaian insiden.

Penelitian ini mengadopsi pendekatan kualitatif dengan melakukan analisis tematik pada 26 laporan post-mortem dari Agustus 2023 hingga Agustus 2024. Temuan menunjukkan bahwa 80% insiden dipicu oleh perubahan internal. Berbagai permasalahan sistemik ditemukan, termasuk kekurangan cakupan pengujian, kelemahan dalam proses deployment dan kontrol perubahan, serta penanganan konfigurasi yang kurang optimal. Selain itu, penelitian ini menemukan bahwa 69% insiden besar tidak memiliki sistem peringatan yang memadai, menyebabkan terlambatnya deteksi dengan Mean Time to Detect (MTTD) selama 82,5 jam. Ketidakefisienan dalam respons insiden, seperti eskalasi yang tertunda dan pemantauan pasca-perbaikan yang tidak cukup, turut berkontribusi terhadap Mean Time to Resolve (MTTR) selama 38,5 jam, yang jauh melampaui standar industri.

Untuk mengatasi tantangan ini, penelitian ini mengusulkan berbagai perbaikan terstruktur dalam manajemen insiden TI. Rekomendasi utama mencakup peningkatan pengujian otomatis, perbaikan prosedur deployment dan tinjauan perubahan, penerapan prosedur eskalasi on-call yang lebih terstruktur, serta standarisasi mekanisme peringatan insiden. Selain itu, pemantauan pasca-perbaikan harus diperkuat guna memastikan efektivitas penyelesaian masalah.

Penelitian ini berkontribusi dalam manajemen insiden TI dengan menyajikan wawasan berbasis data mengenai kegagalan sistemik serta rekomendasi praktis untuk mengurangi frekuensi insiden, meningkatkan cakupan pemantauan, dan mengoptimalkan alur respons insiden. Dengan menerapkan perbaikan ini, PT INUSA dan perusahaan fintech serupa dapat meningkatkan ketahanan sistem, serta meminimalkan dampak bisnis akibat kegagalan sistem TI.

Keywords: Manajemen Insiden IT, Pencegahan Insiden, Deteksi Insiden, Fintech

STRATEGY TO MINIMIZE INFORMATION TECHNOLOGY INCIDENTS AND THEIR IMPACT AT PT INOVINDO NUSAKARYA, A FINTECH COMPANY

By

Faris Arifiansyah

Student ID: 29123055

Master of Business Administration Program

Institut Teknologi Bandung

Approved

Date 22 April 2025

Supervisor

(Dr. Yuanita Handayati)

DECLARATION OF NON-PLAGIARISM

Plagiarism is:

Taking, using, and submitting a work of another, including an idea, writing, or invention, as if it was her or his own.

Plagiarism includes (but not limited to):

1. **quoting verbatim** the work of another without acknowledgement of the source;
2. **paraphrasing** the work of another without acknowledgement of the source;
3. **using** the idea of another without acknowledgement of the source;
4. **submitting** the work of another without identifying clearly who did the work;
5. **colluding** by submitting the work of another as her or his own with consent from the other.

I understand that Plagiarism is wrong, a breach of academic integrity, and against Program, School, and University's Policy and Regulation.

I declare that all material in this Final Project is original, my own work, and does not involve plagiarism.

I have not allowed, and will not allow, another to copy my work with the intention of submitting it as her or his own work.

Name ___Faris Arifiansyah______ Student ID__29123055_______________

Signed _________________________Date ___21 April 2025_______________

FINAL PROJECT GUIDANCE

Unpublished masters' final project is registered and available in the Library of Bandung Institut of Technology, and is open for public, provided that the author owns the copyright in accordance with the intellectual property rights that are applicable in the Library of Institut Teknologi Bandung. Bibliographical references are allowed to be use in a limited manner; however the citation and summarization can only be proceeded upon the author's permission and must include the scientific norm of stating this final project as the source.

This final project has to be cited as:

Arifiansyah, F. (2025): *Strategy to Minimize Information Technnology Incidents and Their Impact at PT Inovindo Nusakarya, a Fintech Company*, Masters' Final Project, Institut Teknologi Bandung.

Reproduction or publication of parts or whole of the final project must be under the consent of the Head of Graduate School, Bandung Institute of Technology.

This final project is dedicated to my beloved family—my wife, my son, my mother, and my brother—who have always supported me.

ACKNOWLEDGEMENTS

Alhamdulillah, all praises to Allah SWT for granting me the health, strength, and perseverance to balance my work and studies, allowing me to complete this thesis. Without His blessings, none of this would have been possible.

To my wife, Ru'fatiani, and my son, Zavier Kaka Alfarezel, thank you for your patience, understanding, and unwavering support. I know that my time with you has been limited due to work and studies, but your willingness to give me the space and focus I needed means the world to me. This achievement is as much yours as it is mine. To my mother and brother, thank you for your endless prayers and emotional support, always being there for me in ways that words cannot fully express.

I am deeply grateful to my thesis advisor, Dr. Yuanita, for always allocating time every week for *bimbingan* sessions, providing insightful feedback, and pushing me to aim higher. Your guidance has been invaluable throughout this journey.

I also want to express my sincere gratitude to my previous managers, Setia (Almarhum) and Azki, who approved and supported my decision to pursue a master’s degree while working. Without their approval and encouragement, I wouldn’t have been able to take this step. To my current manager, Philbert, thank you for your continuous support, regularly checking on my study progress, and ensuring that I had the space to focus on my thesis.

To my colleagues, especially from the Incident Management Team and Site Quality & Reliability Team, thank you for the insightful discussions and for being great partners in analyzing IT incidents. Your input has been incredibly valuable in shaping this research. To my subordinates, I truly appreciate your support in covering for me whenever I was unavailable due to my studies, your reliability allowed me to focus without worry.

To all MBA lecturers, thank you for the knowledge and perspectives you have shared, broadening my understanding beyond my own industry. To my fellow MBA classmates, especially Syndicate 6, I am grateful for all the collaborations, late-night discussions, and shared experiences throughout this journey. It has been a privilege to learn and grow alongside all of you.

This thesis is the result of collective support from so many people, and I am truly blessed to have such an incredible support system. Thank you all.

TABLE OF CONTENTS

LIST OF APPENDICES

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND SYMBOLS

ABBREVIATIONS	Name	Page of initial usage
AI	Artificial Intelligence	28
AIOps	Artificial Intelligence for IT Operations	30
API	Application programming interface	57
BSOD	Blue Screen of Death	1
CD	Continuous Deployment	26
CDE	Continuous Delivery	26
CI	Continuous Integration	26
COS	Cloud Outage Study	28
CPU	Central processing unit	68
Fintech	Financial technology	4
FMEA	Failure Mode and Effect Analysiss	21
FTA	Fault Tree Analysis	21
GMV	Gross Merchandise Value	10
HPA	Horizontal Pod Autoscaler	57
IaC	Infrastructure as Code	27
IDR	Indonesian Rupiah	10
INUSA	Inovindo Nusakarya	4
IP	Internet Protocol	55
IT	Information Technology	1
ITIC	Information Technology Intelligence Consulting Corp.	3
ITIL	IT Infrastructure Library	19
KPI	Key Performance Indicator	79
ML	Machine Learning	23
MR	Merge Request	75
MSME	Micro, small, and medium-sized enterprises	5
MTTD	Mean time-to-detection	15
MTTR	Mean time-to- resolution	15
O2O	Online to Offline	5
OS	Operating System	1
PC	Personal Computer	1
QRIS	Quick Response Code Indonesia Standard	47
RCA	Root cause analysis	9
SDK	Software Development Kit	62
SLO	Service Level Objectives	79
SOP	Standard Operating Procedure	76
TTD	Time-to-detection	14
TTL	Time to Live	54
TTR	Time-to-resolution	14

Chapter I Introduction

I.1 Background

Today, we live in a time where technology has become an essential part of everyday life. Smartphones are now indispensable. It gives us quick access to a wide range of services and information. Mobile apps have completely changed how we interact with the world, from staying connected on social media to managing tasks and enjoying entertainment. At the forefront of this digital revolution are technology companies. These innovative organizations develop and deliver the software, hardware, and services that power our digital lives.

However, despite the convenience and benefits they offer, technology companies, face the persistent challenge of achieving zero incidents. An IT incident refers to an unexpected event or unplanned interruption that disrupts business operational processes or reduces the quality of an IT service (International Organization for Standardization, 2018). The complex nature of modern technology systems, along with the constant introduction of new products and features, make it more difficult to eliminate disruptions entirely. Disruptions can have severe consequences, such as financial losses, reputational damage, and operational disruptions.

In the past years, numerous IT incidents have highlighted the importance of service reliability. For example, the CrowdStrike incident in July 2024 caused widespread disruptions as thousands of commercial flights globally were canceled due to a software bug pushed to Windows Update by CrowdStrike that triggered the blue screen of death (BSOD) in Windows OS PCs. This incident resulted in estimated losses between US$300 million and US$1 billion (Guy Carpenter, 2024), and the company's stock price fell more than 15% in the following days.

Another notable incident occurred in October 2021 when Facebook and its affiliated services, such as WhatsApp and Instagram, experienced a global outage lasting nearly 6 hours. The outage was caused by a misconfiguration, leading to a significant drop in Facebook's share price by nearly 5%. It is estimated that the company lost approximately US$79 million in ad revenue during the outage (Lee, 2021). These incidents have demonstrated the significant consequences of disruptions, including financial losses, reputational damage, and customer dissatisfaction (Chen, et al., 2020).

An IT incident can vary in scope and impact, ranging from minor service degradations to full-scale outages. While an incident refers to any event that disrupts normal IT operations, an outage is a severe form of incident where a system or service becomes entirely unavailable (SolarWinds, 2025). Figure I.1 shows data from 2019 to 2022 on the number of publicly reported outages. While advancements in technology, software, and physical redundancy have helped organizations reduce the severity of individual outages as can be seen by the decline in "Severe" and "Significant" classifications, the overall frequency of outages remains high, showing no sign of decreasing. Each year, a portion of organizations still experience impactful disruptions, with nearly one-third of outages consistently rated as "Minimal" and another third as "Negligible." This trend suggests that despite improvements in handling the impact of outages, the fundamental risk of disruptions still persists.

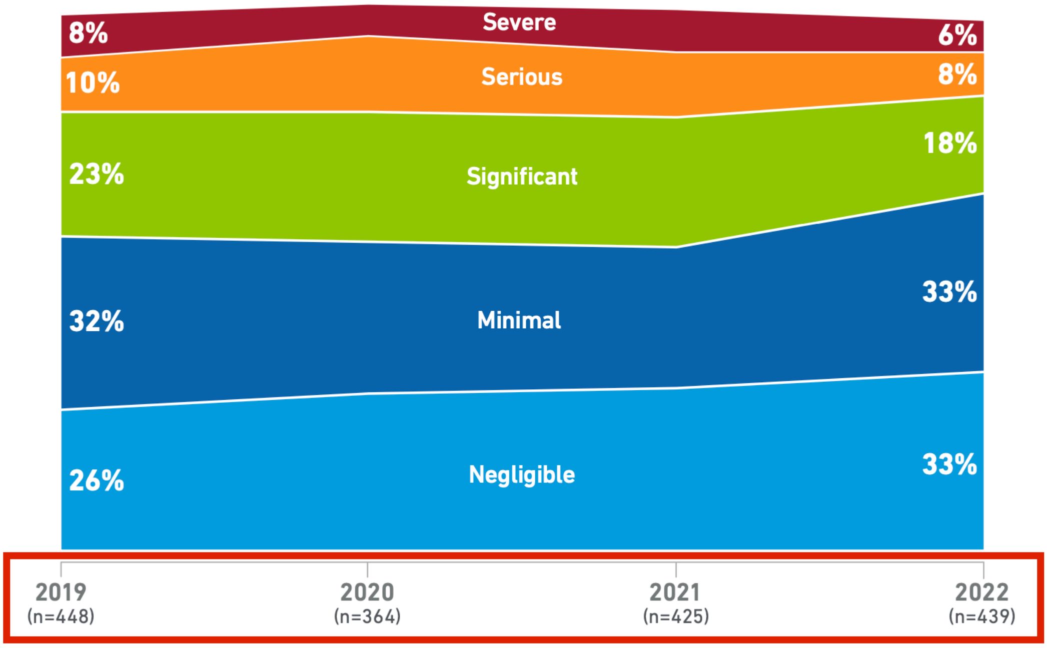


Figure I.1 Proportion of outages classified as significant, serious, or severe from 2019 to 2022 (Lawrence & Simon, 2023)

From an industry perspective, Figure I.2 shows the financial impact of server downtime across various sectors. According to data collected in 2017 by Information Technology Intelligence Consulting Corp. (ITIC), the banking and finance industry, for example, incurs an estimated loss of $9.3 million per hour of downtime. This emphasizes the critical importance of service reliability and effective incident management, especially in financial services which have a heavy reliance on IT infrastructure.

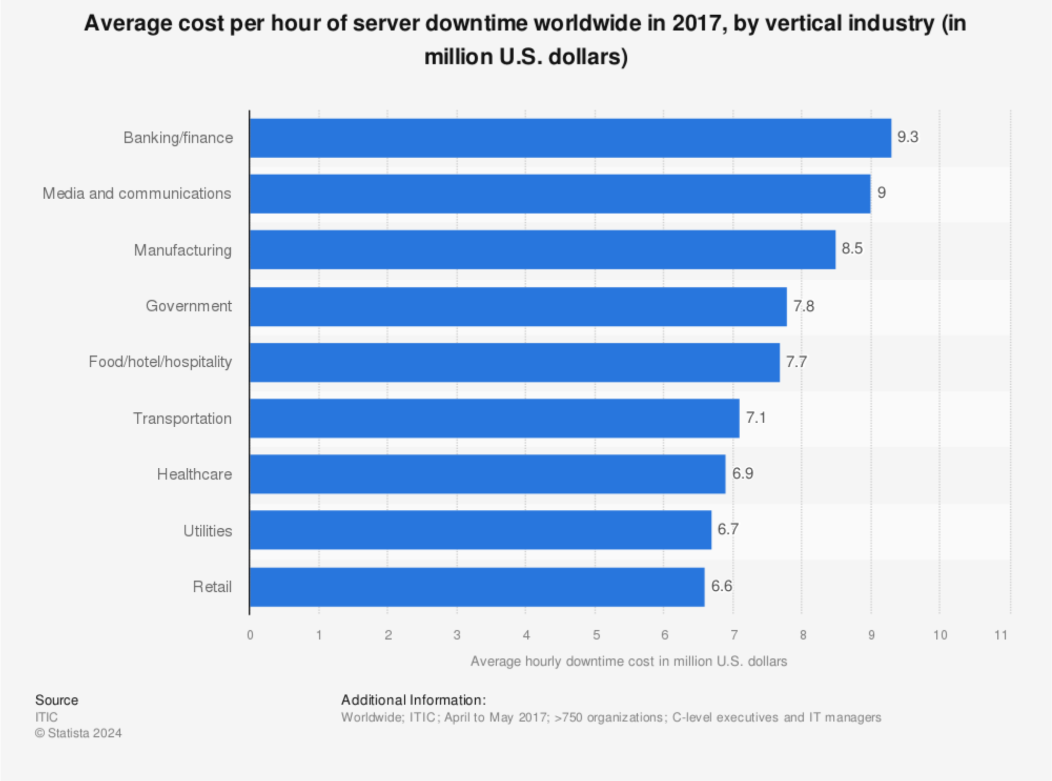


Figure I.2 Cost per hour of server downtime worldwide in 2017, by vertical industry (ITIC, 2017)

As digital financial services continue to evolve, fintech companies have emerged as key players in the financial sector. They leverage technology to provide innovative solutions. They offer a wide range of solutions, including digital payments, lending, wealth management, and insurance. By combining traditional financial services with cutting-edge technology, fintech companies aim to make financial transactions more accessible, efficient, and convenient. Like traditional banks, fintech platforms are highly dependent on IT systems. It makes them equally susceptible to the risks associated with service disruptions.

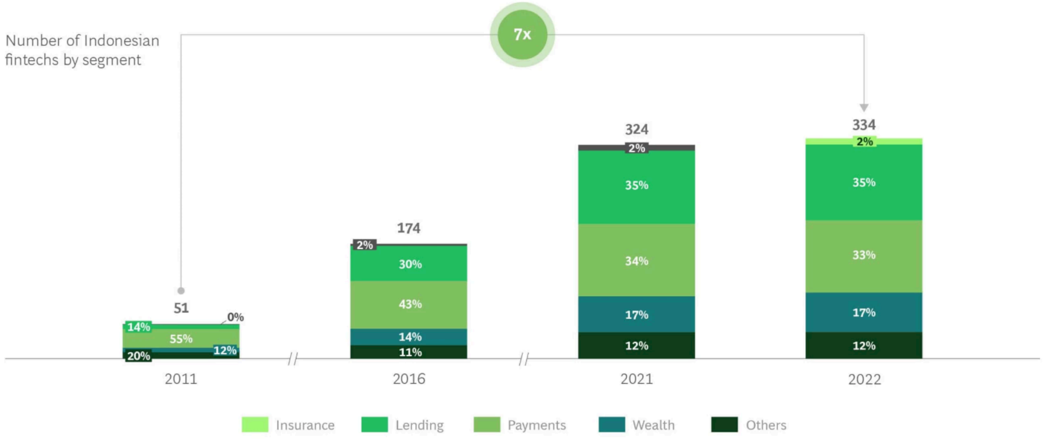


Figure I.3 Indonesia Fintech Industry Growth (Kumar, et al., 2023)

In Indonesia, the fintech industry has experienced remarkable growth over the past decade. As shown in Figure I.3, this sector has expanded exponentially, with a sevenfold increase from 2011 to 2022. PT Inovindo Nusakarya (INUSA) is one of the players in this industry, offering a comprehensive digital platform through its mobile application, which provides a wide range of financial services. These services now represent the company's primary revenue drivers. However, PT INUSA has faced significant challenges, experiencing a high volume of incidents over the past year. These incidents have resulted in considerable financial losses and negatively affected the company's reputation. This thesis will explore IT incidents at PT INUSA, investigating their root causes and examining best practices in incident management, with the goal of developing strategies to reduce both the frequency and impact of such events.

I.2 Company Profile

This section presents an overview of the company's background. It covers the company’s mission, core values, organizational structure, and other relevant information. The Engineering Division is also presented in a dedicated section to understand more about its responsibilities and the relation with incident handling.

I.2.1 PT Inovindo Nusakarya (INUSA)

PT Inovindo Nusakarya (INUSA) is a technology company in Indonesia which was founded in 2010. It was launched as an e-commerce platform with the goal to become the largest in the country. PT INUSA experienced rapid growth and then diversified its offerings. It expanded and introduced several new platforms, such as an online-to-offline (O2O) service, an investment platform, and a gaming platform. In 2020, the company committed to focus more on achieving profitability through sustainable business practices. The O2O platform showed strong growth potential and since then, it becomes the company's primary revenue driver. The platform is accessible via a mobile application available on the Google Play Store. It specifically designed for micro, small, and medium-sized enterprises (MSMEs) in Indonesia. It is commonly used by kiosk owners (*warung*), phone credit agents (*agen pulsa*), and other small businesses to provide various digital services to customers who visit their physical stores. Through the app, MSMEs can sell mobile phone credits, pay bills, and process financial transactions. These features make the app a key tool for MSMEs to expand their business and service offerings. As of September 2024, the platform has a total of 18 million registered users. Table I.1 provides a list of the features available on the platform.

Table I.1 Features Offered in PT INUSA's Platform

No	Feature Name	Description
1	Kirim Uang	Allows MSMEs to help customers transfer money to bank accounts directly through the app.
2	Pulsa	Allows MSMEs to sell mobile phone credit (top-ups) to customers.
3	Paket Data	Enables MSMEs to offer data packages for mobile phones, allowing customers to purchase internet data.
4	Token Listrik	Allows MSMEs to sell prepaid electricity tokens, enabling customers to purchase and recharge their prepaid electricity accounts directly through the app.
5	Top Up E-wallet	Allows MSMEs to facilitate the loading of funds into various e-wallets such as OVO, Gopay, and DANA for their customers, enabling easy digital transactions and payments directly through the app
6	Top Up E-money	Enables MSMEs to help customers add funds to their Mandiri e-money and BCA Flazz accounts
7	Top Up Game	Allows MSMEs to sell in-game credits or game vouchers for various online games, enabling customers to easily purchase and recharge their gaming accounts directly through the app.

No	Feature Name	Description
8	Bayar Virtual Account	Allows MSMEs to assist customers in making payments to various merchants or service providers through virtual accounts.
9	Tagihan Listrik PLN	Allows MSMEs to facilitate the payment of electricity bills for customers who use PLN (Perusahaan Listrik Negara).
10	PDAM	Allows MSMEs to assist customers in paying their water bills for PDAM (Perusahaan Daerah Air Minum).
11	Angsuran Kredit	Allows MSMEs to assist customers in making installment payments for their loans directly through the app.
12	Telkom/IndiHome	Enables MSMEs to facilitate the payment of Telkom and IndiHome services for their customers.
13	BPJS Kesehatan	Allows MSMEs to help customers pay their health insurance premiums to BPJS Kesehatan.
14	BPJS Ketenagakerjaan	Enables MSMEs to assist customers in paying their employment insurance premiums to BPJS Ketenagakerjaan conveniently via the app.
15	E-Samsat & SIGNAL	Allows MSMEs to facilitate the payment of vehicle taxes through the app, making it easier for customers to manage their vehicle-related payments.
16	Penerimaan Negara	Enables MSMEs to assist customers in making payments for various government revenue services, including passport fees, income tax, and customs duties.
17	Pajak PBB	Allows MSMEs to help customers pay their property taxes (Pajak Bumi dan Bangunan) conveniently.
18	Pulsa Pascabayar	Allows MSMEs to assist customers in paying postpaid mobile phone bills directly through the app
19	Zakat	Enables MSMEs to facilitate the payment of zakat (charitable donations) for customers.

To guide its culture and operations, PT INUSA upholds the following core values:

1. Collaboration and Mutual Support

   The company emphasizes the importance of teamwork. It is where individuals across all levels work together to achieve common goals. This spirit of collective effort ensures that challenges are tackled more efficiently.

2. Empowering Customers

   A primary focus of the company is helping its customers grow. By providing the right tools, services, and support, the company enables its customers to grow and succeed in their own businesses.

3. Data-Driven Decision Making

   The company believes that decisions should be based on data and facts, not emotions or assumptions. This helps ensure that actions and plans are made and decided using real information so it can produce better results.
4. Commitment and Accountability

   Employees are encouraged to take responsibility for their work and do their best to achieve high-quality results.
5. Simplicity in Solutions

   The company values simple and clear processes because the best solutions are often the easiest to understand. Keeping things simple also helps improve innovation and efficiency.
6. Positive and Enjoyable Work Culture

   The company promotes a work environment that is both productive and enjoyable. By creating a fun and engaging atmosphere, employees can work better together, be more creative, and feel more satisfied with their jobs.

To provide a clearer understanding of PT INUSA's organizational structure, Figure I.4 presents a high-level overview of the structure, with the Engineering Division presented in detail, as it is the division who is responsible for incident management and prevention strategies. The following section explores its structure and responsibilities in greater detail.

I.2.2 Engineering Division

As a technology company, one of the primary and largest functions is the engineering division. The Engineering Division at PT INUSA has a critical role in supporting the company's technological aspects. The engineering team is responsible for the development, maintenance, and doing continuous improvement of the platforms that provide PT INUSA's services. The activity consists of building scalable solutions, ensuring platform or system reliability, and implementing new features that align with business needs.

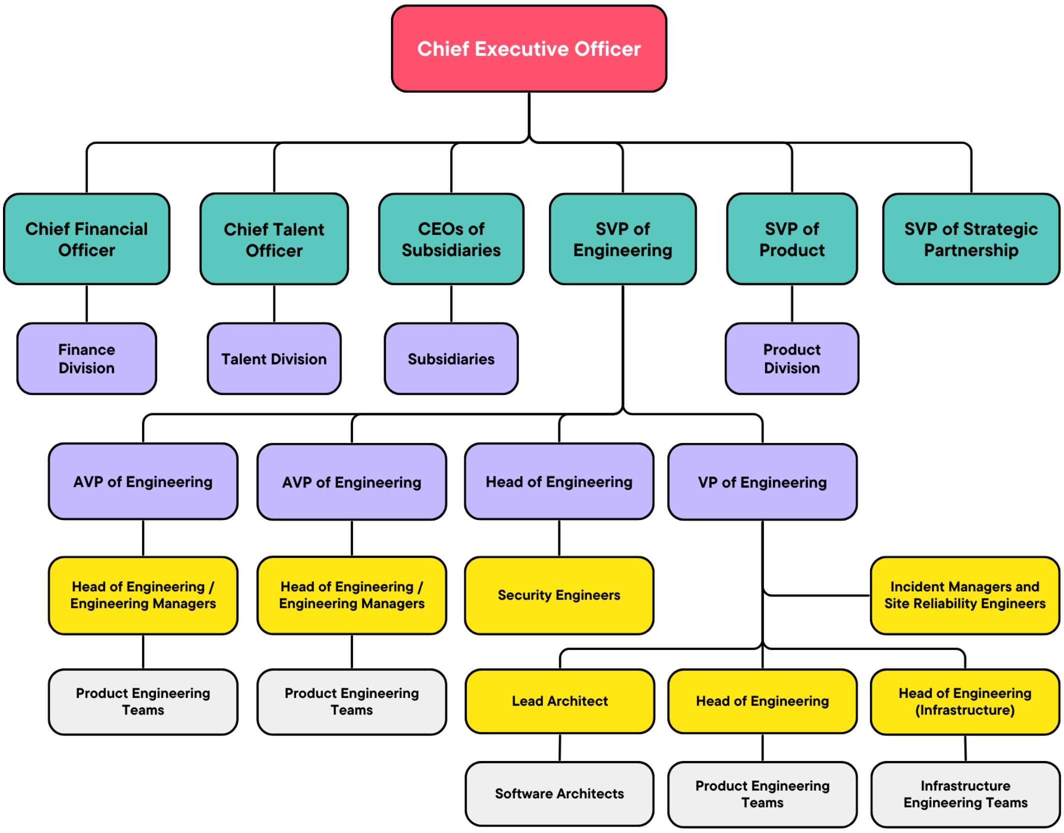


Figure I.4 PT INUSA’s Organizatioal Structure

The division is organized into multiple specialized squads. Each squad focus on different aspects of the platform. As of September 2024, the engineering division consists of around 250 people, led by the Senior Vice President (SVP) of Technology. There are a total of around 30 squads. These squads usually consist of backend engineers, frontend developers, and mobile app developers. Additionally, infrastructure and DevOps teams also work closely with them to ensure that deployment, scalability, and system performance meet the company's high standards.

Each squad is given a high level of autonomy but still closely aligned with the company’s overall product roadmap. Agile methodologies, such as Scrum, are utilized to ensure that the teams are flexible and responsive to changes of business requirements. Regular sprints and retrospectives are conducted to enable continuous evaluation and improvement of processes. These are the primary responsibilities of the Engineering Division:

1. Platform Development
   Designing and implementing features for both the O2O platform and other key products such as the gaming and investment platforms.
2. System Maintenance
   Monitoring the stability and performance of the platforms. Responding quickly to any incidents to ensure minimal disruption to users.
3. Innovation and Scalability
   Developing scalable solutions that can support the rapid growth of the user base. As of September 2024, it already exceeded 18 million registered users.
4. Security
   Ensuring that all systems are secure and compliant with relevant regulations.

The engineering team has become a very important group that help PT INUSA's transition from a traditional e-commerce platform to a robust O2O business. The successful development and deployment of the O2O platform has become one of its major achievements as it can serve millions of MSMEs across Indonesia. This platform provides a wide range of services that MSMEs can use to offer their offline customers.

In addition to developing the platform, the Engineering Division is also responsible for incident management to minimize disruptions. To handle incidents effectively, it requires collaboration between Incident Managers, Site Reliability Engineers (SREs), Product Engineering Teams, and Infrastructure Engineering Teams. Incident Managers take the lead in coordinating people to handle incident when an it occurs. They make sure that the right teams are involved. SREs focus on monitoring system health and help product engineering team in conducting root cause analysis (RCA). Product Engineering Teams are the owners of their respective services, which means they are responsible for resolving incidents and conducting post-mortems to learn from failures. Lastly, Infrastructure Engineering Teams work behind the scenes to make sure the platform stays reliable. They set up automated failovers, real-time monitoring, and capacity scaling to reduce the risk of major outages. They are also available to help mitigating incident, especially if the issue is related to infrastructure.

I.3 Business Issue

In the fast-paced environment of PT INUSA, it is crucial to maintain the high level of system reliability and availability to ensure it can satisfy users' needs and business stakeholders. In the past few years, the company has experienced many incidents that affect its platforms. Some of these caused financial losses and potentially have a negative long-term effect on customer trust and retention.

One major incident happened on July 23, 2024, when a critical issue in the database server of the *Token Listrik* product caused a service disruption that lasted about 1.5 hours. Although this outage only affected *Token Listrik* product, its impact was significant. The disruption led to an estimated Gross Merchandise Value (GMV) loss of over IDR 500 million, as successful *Token Listrik* transactions dropped by more than 90% compared to normal levels. Figure I.5 shows this sharp decline, which started at around 17:00 and ended at around 18:30.

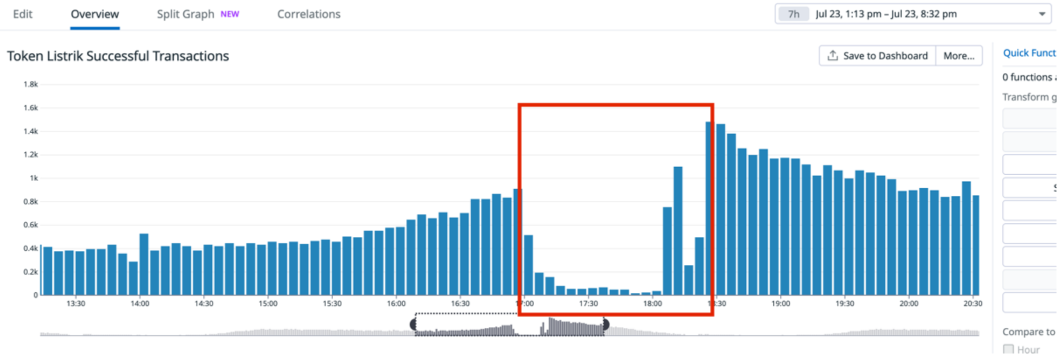


Figure I.5 An Incident Sample at PT INUSA: *Token Listrik* Transaction Drop.

However, the financial loss was not the only impact. The service disruption also caused frustration among users. This was reflected in a surge of customer complaints during the outage. The number of support tickets escalated sharply throughout the incident, as shown in Figure I.6. It highlights the immediate consequences of service disruptions on customer trust and loyalty.

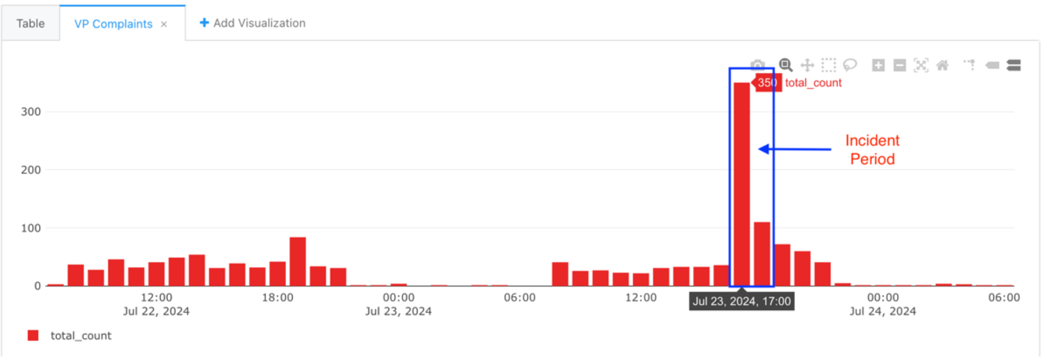


Figure I.6 Increase in Support Tickets in the Digital Product Category During Incidents

In the past year (August 2023 – August 2024), over 200 incidents occurred at PT INUSA, leading to significant financial consequences. The total Gross Merchandise Value (GMV) loss was reaching 94 billion rupiahs, with an estimated revenue loss of around 2.6 billion rupiahs. These figures highlight the need for the company to address incidents proactively and treat these issues with the seriousness they deserve.

PT INUSA classifies incidents into two categories: major and minor. A major incident is defined as one that results in a financial impact exceeding 25 million rupiahs. From the total of 206 incidents recorded between August 2023 and August 2024, 62 were classified as major incidents. Notably, these major incidents accounted for more than 99% of the total financial loss (see Table I.2).

Table I.2 Incidents Summary

	Major Incident	**Total Incident**
Number of Incident	62	206
GMV Loss	93,397,230,891	93,725,901,138
Revenue Loss	2,555,722,943	2,617,410,730

GMV represents the total value of transactions occurring on the platform, and a GMV loss reflects the volume of potential sales that were prevented due to the incident. This figure underscores the extent of market opportunities missed and can also indicate potential erosion of customer trust. Revenue loss, meanwhile, captures the actual income forfeited directly because of the incident. Unlike GMV, revenue has a direct effect on the company’s financial health and profitability.

The occurrence of incidents has also been unpredictable. Figure I.7 displays the monthly distribution of major incidents over the past year, revealing no discernible pattern or downward trend. This unpredictability underscores the importance of paying close attention to incident management and implementing measures to reduce future occurrences.

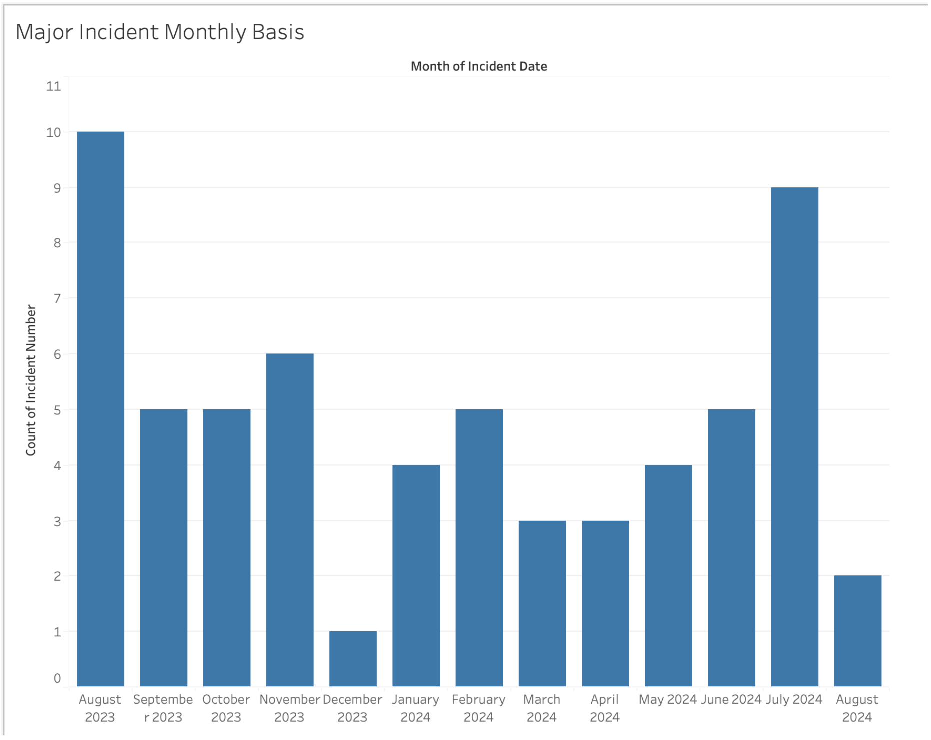
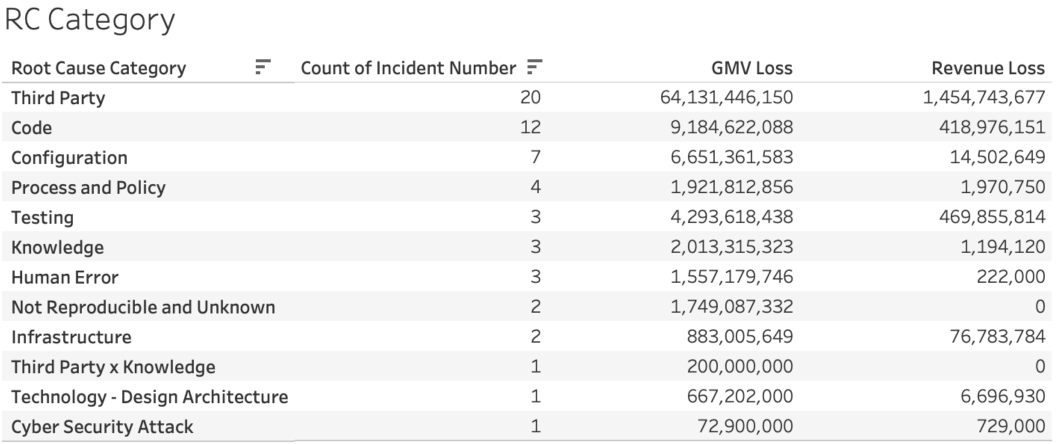


Figure I.7 Monthly Major Incidents

PT INUSA has established an incident management process, one of whose key activities is the post-mortem review. After an incident is mitigated, the squad responsible for the affected service is required to compile a post-mortem document. This document typically includes an incident summary, financial impact, the incident trigger, detection and mitigation timeline, root cause analysis, and lessons learned. A few days after an incident, a post-mortem review session is conducted, facilitated by the incident manager. At the conclusion of the session, the incident manager, in agreement with all relevant engineering stakeholders, determines the root cause category of the incident. Figure I.8 illustrates the root cause categories for major incidents over the past year.



RC Category

Root Cause Category	Count of Incident Number	GMV Loss	Revenue Loss
Third Party	20	64,131,446,150	1,454,743,677
Code	12	9,184,622,088	418,976,151
Configuration	7	6,651,361,583	14,502,649
Process and Policy	4	1,921,812,856	1,970,750
Testing	3	4,293,618,438	469,855,814
Knowledge	3	2,013,315,323	1,194,120
Human Error	3	1,557,179,746	222,000
Not Reproducible and Unknown	2	1,749,087,332	0
Infrastructure	2	883,005,649	76,783,784
Third Party x Knowledge	1	200,000,000	0
Technology - Design Architecture	1	667,202,000	6,696,930
Cyber Security Attack	1	72,900,000	729,000

Figure I.8 Major Incidents by Root Cause Category

The leading root cause category is third party. It refers to issues caused by external providers or partners, which PT INUSA has limited control to it. The next most significant categories are code, configuration, process and policy, and testing. They represent areas that are within the company’s direct control. Code refers to bugs or limitations in the application that hinder functionality. Configuration issues arise from incorrect or inappropriate system settings. Process and policy issues arise from gaps in the processes or policies in place, including the lack of adequate coverage, as well as violations or misapplications of internal procedures. Testing deficiencies occur when systems are not properly tested before deployment. It unintentionally produces unforeseen issues in production. The complete definitions for other root cause categories can be found in Appendix A.

Although the company has an incident management process, the data shows that further improvements are needed, especially in addressing incidents at a systemic level rather than on a case-by-case basis. Therefore, this research will focus on the key root causes that PT INUSA can directly control and address, which are, code, configuration, process & policy, and testing. Understanding these areas and their impact on incidents is crucial to improve the company's incident management strategies and reduce future disruptions.

In addition to understanding the root causes of incidents, another critical issue lies in the company's ability to detect and respond to them effectively. Even though monitoring and alerting mechanisms already in place, many incidents still experienced a delayed detection and slow resolution. Table I.3 presents an overview of Time-to-Detection (TTD) and Time-to-Resolution (TTR) for major incidents at PT INUSA. TTD represents the duration between the start of an incident and its detection, while TTR represents the time taken to resolve the issue after it is detected.

Table I.3 Incident Detection and Resolution Times

Incident Number	TTD (minutes)	TTR (minutes)	Total Incident Duration (minutes)
2024080704	0	14	14
2024080501	5,583	5,110	10,693
2024071101	24,529	1,371	25,900
2024070301	870	370	1,240
2024070202	11	78	89
2024070102	148	311	459
2024052901	6,288	30,519	36,807
2024042201	11,429	10,085	21,514
2024042101	254	6	260
2024031903	7,401	49	7,450
2024022901	0	175	175
2024021901	5,464	306	5,770
2024020601	7,260	300	7,560
2024013001	815	30	845
2024011801	40,839	8,664	49,503
2023112401	0	96	96
2023110301	348	60	408
2023102601	1,090	14	1,104
2023101901	31	30	61
2023101702	8,562	1,939	10,501
2023101701	790	170	960
2023092202	17	173	190
2023092102	11	28	39
2023091301	16	12	28
2023081501	6,859	184	7,043
2023081001	75	60	135
Average	**4,950**	**2,314**	**7,263**

As shown in the table, the Mean Time to Detect (MTTD) across major incidents is 4,950 minutes (82.5 hours), while the Mean Time to Resolve (MTTR) is 2,314 minutes (38.5 hours). These numbers are substantially higher than industry standards. According to a 2023 survey by New Relic, 44% of organizations detect high-impact incidents within 30 minutes, and only 21% exceed 60 minutes. Similarly, 60% of organizations resolve incidents within 30 minutes, and only 34% take over an hour. When compared to these benchmarks, PT INUSA's MTTD is over 80 times longer than industry best practices, and its MTTR significantly exceeds acceptable thresholds. This highlight a critical gap in PT INUSA’s incident management process—not only do incidents occur frequently, but once they happen, they are neither detected nor resolved in a timely manner. These inefficiencies amplified the impact of system failures. Therefore, this research, aims to analyze both the systemic causes of incidents and the inefficiencies in incident detection and response. Addressing these challenges is essential to reducing business disruptions and improving the company’s overall resilience.

I.4 Research Questions and Research Objectives

Given the significant impact of incidents on financial performance and operational stability, this research aims to explore the systemic factors contributing to these disruptions. The study focuses on incidents arising from code, configuration, process and policy, and testing, as these are within the direct control of the organization. To address this, the research will answer the following key questions:

1. What are the primary causes of IT incidents, particularly in the areas of code, configuration, processes, and testing?
2. What factors contribute to delays in incident detection and response, and how do they impact system disruptions?
3. What strategies can be implemented to improve IT incident prevention, detection, and response?

To address these research questions, this study has the following objectives:

1. To identify key patterns and analyze the common causes of incidents related to code, configuration, process, and testing. This objective aims to explore the underlying factors that contribute to recurring incidents.
2. To identify factors that contribute to delays in incident detection and response. The goal is to analyze how these factors impact system disruptions and incident resolution times.
3. To propose strategies for improving IT incident management. This includes preventative strategies, enhanced detection mechanisms, and optimized response workflows. These recommendations aim to minimize incident occurrence and reduce detection and resolution times.

I.5 Research Scope and Limitation

This research focuses on analyzing the IT incidents experienced by PT INUSA during the period of August 2023 until August 2024. Incidents that are analyzed are related to code, configuration, processes and policies, and testing. These root cause categories were selected because they are controllable, and their frequency and impact were also significant to the company's business. These are the scope of this research:

1. Investigate systemic failure patterns and identify the primary causes of IT incidents within the selected categories.
2. Analyze gaps in incident detection and response mechanisms to understand factors that contribute to delayed detection and prolonged resolution times.
3. Propose practical recommendations to enhance incident prevention, detection, and response.
4. Draw conclusions based on PT INUSA's internal incident data, especially major incidents that resulted in substantial financial losses or service disruptions.

The limitations of this research are as follows:

1. Incidents caused by third-party providers are excluded from the analysis, as PT INUSA has limited control over these issues. Even though they can have

a significant impact, this research focuses only on areas that are directly manageable by the company.

2. The analysis is based on data from a single year (August 2023 – August 2024), which may not fully capture long-term trends or fluctuations over a broader timeline.
3. The findings and recommendations in this study are specific to PT INUSA's operational context and may not be directly applicable to organizations with different infrastructures or incident management workflows. However, the insights on systemic failure patterns and incident response inefficiencies may still offer relevant takeaways for similar organizations operating in technology-driven environments.
4. This research will not cover incidents unrelated to IT (such as those involving physical infrastructure or human resources) and will concentrate solely on technology-related disruptions.
5. This research will provide practical recommendations for PT INUSA based on the analysis. However, the scope is limited to the development of implementation plans. The actual implementation and subsequent quantitative evaluation of the recommendations are not included in this study result.

Despite these limitations, the research will provide valuable insights into improving PT INUSA's incident management processes and preventative measures. To ensure the study is well-structured, it follows the standard thesis structure prescribed by the Bandung Institute of Technology:

CHAPTER 1 INTRODUCTION

Chapter 1 introduces the research context. It defines the problem, outline the objectives and setting the scope and limitations.

CHAPTER 2 LITERATURE REVIEW

Chapter 2 provides an in-depth discussion of relevant theories and concepts based on academic books, journals, and previous research.

CHAPTER 3 RESEARCH METHODOLOGY

Chapter 3 explains the research design, data collection methods, and data analysis techniques that are used in this study. It ensures that the research process is clear, systematic, and reliable.

CHAPTER 4 RESULTS AND DISCUSSION

Chapter 4 presents the results of a comprehensive analysis of IT incidents at PT INUSA. The key failure patterns, detection gaps, and response inefficiencies were identified in this chapter. Based on these findings, this chapter also provides recommendations to improve incident prevention, enhance detection mechanisms, and improve response strategies to reduce future disruptions.

CHAPTER 5 CONCLUSION AND RECOMMENDATION

Chapter 5 summarizes the key findings of the study, presents conclusions, and offers recommendations to enhance incident management strategies in the PT INUSA's Engineering Division. It also discusses potential areas for future research.

Chapter II Literature Review

II.1 Theoretical Foundation

This section provides an overview of key concepts, definitions, and theories that form the foundation of this research. It is collected from relevant literature on IT incident management, root cause analysis, and IT change management as part of efforts to reduce and prevent incidents.

II.1.1 Incident Management in IT Systems

Incident management is a key process in IT service management that focuses on quickly identifying, analyzing, and resolving disruptions in IT systems. It ensures that issues are addressed efficiently to minimize downtime and reduce business impact. According to the IT Infrastructure Library (ITIL) framework, an incident is defined as "an unplanned interruption to an IT service or a reduction in the quality of an IT service" (AXELOS, 2019). The primary objective of incident management is to restore normal service operations as quickly as possible while minimizing the negative impact on business operations. In industries where service availability is directly linked to revenue and customer satisfaction, such as finance, healthcare, and e-commerce, efficient incident management becomes essential. Downtime or service degradation can not only lead to financial losses but also harm an organization’s reputation, regulatory compliance, and overall customer trust (Lawrence & Simon, 2023).

Effective incident management includes several key components. Figure II.1 shows ITIL Incident Management Process. First is incident detection and recording. This is often achieved through automated monitoring systems and helpdesk reports. Rapid detection enables early intervention, which is critical for minimizing downtime. After detection, incidents are prioritized and categorized based on their urgency and potential impact. For example, incidents affecting core business operations or large user groups are prioritized higher, ensuring that resources are directed to issues that could have the most severe consequences (Bashir & Soomro, 2012).

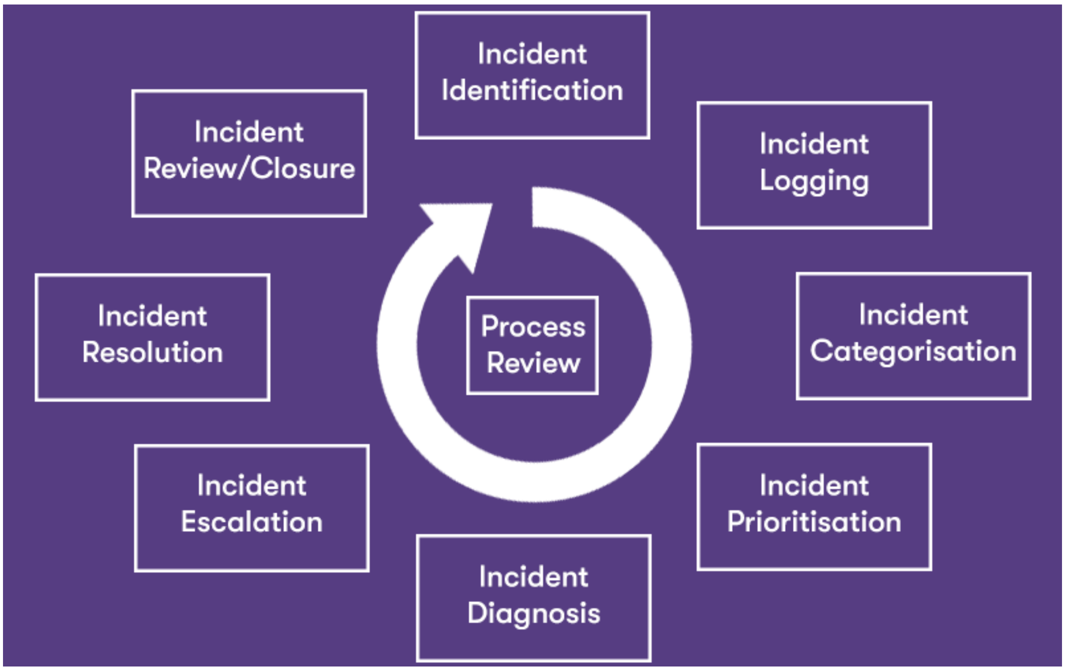


Figure II.1 Incident Management Process (Lawless, ITIL Incident Management - Everything You Need To Know, 2023) (Richard, Gaol, Warnars, Abdurachman, & Soewito, 2019)

When incidents require specialized knowledge or cannot be resolved at the initial point of contact, they are escalated to higher-level support teams or specialists. This escalation process ensures that complex issues receive the expertise necessary for effective resolution (Atlassian, 2024). After an incident is resolved, the case is formally closed in the incident management system and the affected user is promptly informed of the outcome. In addition, incident details are communicated to key stakeholders to keep them updated on service performance and emerging incident trends. For major incidents with considerable business impact, a post-incident review is conducted to evaluate the investigation process, assess the effectiveness of the resolution, and pinpoint areas for future improvement. While the primary goal of incident management is to respond swiftly, root cause analysis frequently follows to identify underlying factors and prevent recurrence. Root cause analysis enables IT teams to apply permanent fixes rather than recurring temporary solutions, reducing the likelihood of repeated incidents.

With the growing complexity of digital ecosystems, incident management in modern IT has become more challenging. Cloud computing, microservices architectures, and extensive cross-platform integrations add new layers of complexity, complicating both incident detection and response (Chen, et al., 2020). Thus, a continuous improvement approach is crucial. Organizations must learn from past incidents, refine their management practices, and adapt to the evolving technology landscape.

As technology advances, incident management has shifted from only reactive measures to include proactive strategies for incident prevention and mitigation. Google’s experience highlights the value of a proactive incident management strategy that is scalable, regularly exercised, and designed to streamline response efforts. This approach has significantly reduced Google’s mean time to recovery and has created a less stressful environment for staff when managing unexpected issues (Beyer, Jones, Petoff, & Murphy, 2016).

II.1.2 Root Cause Analysis

One of the key activities of incident management is the root cause analysis (RCA). This activity is very important to ensure that the cause of the incident can be found and fixed hence the same issue would never happen again. Root cause analysis itself is a process aimed at identifying the underlying causes of a problem so that corrective actions can be taken. Corrective actions directly target these root causes rather than only the symptomps (Okes, 2009).

There are various techniques to conduct a root cause analysis. Some of the most popular techniques are the 5 Whys, Fishbone Diagram, Pareto Chart, Failure Mode and Effect Analysis (FMEA), and Fault Tree Analysis (FTA). The "5 Whys" Technique is an iterative method involves asking "why" multiple times (typically five) to delve deeper into the cause-and-effect relationships underlying a problem. Each answer forms the basis for the next question, progressively leading to the root cause. This technique is valued for its simplicity and effectiveness in identifying fundamental issues (Reid & Smyth-Renshaw, 2012). Another popular technique, the Fishbone Diagram, is a tool created by Kaoru Ishikawa designed to systematically organize and display the relationships between various potential causes of a problem. By encouraging structured thinking, it helps problem-solving teams focus on possible root causes, enabling clearer discussions and more efficient efforts to identify the actual source or sources of an issue. Here are the steps to create the fishbone diagram (Juran & Feo, 2010):

1. Clearly identify the outcome (the Y) that needs to be investigated to find the root cause.
2. Position the effect or symptom being analyzed on the right side, enclosed in a box. Draw a central line (spine) pointing towards it.
3. Use brainstorming or a structured, logical approach to identify the potential causes (the Xs).
4. For each major category of possible causes (typically two to five), place them in a box and connect them to the central spine with lines at an approximate 70-degree angle.
5. For each primary category, identify specific potential causes and place them on the horizontal lines branching from the spine.
6. Identify sub-causes for each of the main causes already listed and add them to the diagram.
7. Keep adding causes to the branches until each one reaches a potential root cause.
8. Review the logical sequence of each cause-and-effect chain to ensure it makes sense (e.g., a flat tire caused the car to swerve, the nail caused the flat tire, a person left the nail on the driveway).
9. Ensure the diagram is complete by checking if all potential causes have been explored.

In IT incident management, RCA is integral to the problem management process. After an occurrence of an incident, immediate actions focus on restoring service to minimize impact. Subsequently, RCA is conducted to determine the underlying cause of the incident. This involves collecting data related to the incident, analyzing system logs, and consulting with stakeholders to reconstruct the sequence of events that lead to the failure (Saha & Hoi, 2022). Insights gained from RCA inform the development of corrective actions, such as process improvements, system upgrades, or training programs, with the purpose to prevent similar incidents in the future.

Recent advancements have seen the integration of Machine Learning (ML) techniques into RCA processes. ML-enhanced RCA can automate the identification of causal relationships in high-dimensional datasets. It reduces reliance on manual expertise and expediting incident resolution. For instance, ML algorithms can process vast amounts of structured and unstructured data, including log files and sensor readings, to identify root causes more effectively (Katragadda, Peddinti, Pandey, & Tanikonda, 2021).

II.1.3 IT Change Management

PT INUSA is a software-defined business. A software-defined business refers to an organization where the creation of value whether through its processes, products, or business models relies heavily on software and services (Alt, et al., 2020). In a software-defined business, updates and improvements to products are deployed on a daily, weekly, or monthly basis. In the case of PT INUSA, changes or updates to the production environment happen in a daily basis. It indicates how fast the development pace in the company. However, despite the benefits they offer, the risk is also higher. Studies show that more than 70% of IT incidents were caused by a change (Beyer, Jones, Petoff, & Murphy, 2016) (Güven & Murthy, 2016). That is why understanding the IT change management processes is very crucial on preventing IT incidents. This section will provide the explanation about IT change management processes.

II.1.3.1 Change Control

A change refers to any addition, alteration, or removal of anything that may directly or indirectly impact the delivery of services. The scope of change control may vary between organizations but typically it includes all IT infrastructure, applications, processes, documentation, and anything else that might impact a product or service (directly or indirectly). At PT INUSA, change often happens to applications and IT infrastructure. Change control is focused on changes in product or service rather than the people aspects (organizational change management).

According to ITIL management practices, change control requires a careful balance between implementing beneficial changes that add value and safeguarding customers and users from potential negative impacts. All proposed changes should be evaluated by qualified individuals that are capable of assessing both risks and benefits. Changes must be authorized prior to deployment but it should not cause unnecessary delays in the process (AXELOS, 2019). The person or groups who authorize a change is called a change authority. ITIL divides changes into three types:

1. Standard changes

   These are low-risk changes. When a standard change procedure is created or updated, it must undergo a complete risk assessment and authorization, like any other type of change. However, this assessment does not need to be repeated for every time the standard change is implemented. It is only required if the procedure itself is modified.

2. Normal changes

   These changes follow a structured process that involves scheduling, assessment, and authorization. The roles responsible for these steps are determined by change models based on the type and risk level of the change. Low-risk normal changes are often handled by individuals with the authority to make quick decisions. It often uses automation to speed up the process. Higher-risk changes may need approval from senior management. These changes usually start with a change request, which can be created manually or automated in organizations using continuous deployment pipelines.

3. Emergency changes

   These changes are those that must be implemented immediately, such as resolving critical incidents or applying security patches. They are usually not included in the regular change schedule, and the assessment and approval process are accelerated to minimize delays. Ideally, emergency changes should follow the same testing, evaluation, and approval steps as normal changes. However, some steps, like documentation or extensive testing, may be deferred due to time constraints. These changes are usually overseen by a separate change authority. It often composed of senior managers who understand the associated business risks.

II.1.3.2 Deployment Management

The goal of deployment management is to move new or updated hardware, software, documents, processes, or other components into live environments. It can also handle deploying these components to other environments, like testing or staging, to ensure everything works as expected before going live. In some organizations, the term "provisioning" is used to refer to deployment of infrastructure, while "deployment" is only used to refer software deployment. However, in this context, "deployment" refers to both infrastructure and software (AXELOS, 2019).

There are various deployment methods available, and organizations often adopt a mix of these approaches. The choice depends on their specific needs, such as the nature of their services, release characteristics, and the impact of the changes.

1. Phased Deployment

   In this approach, the new or modified components are gradually deployed to some parts of the production environment, such as users in a specific criterion or located in certain region. This process is repeated as necessary until the deployment is fully rolled out.

2. Continuous Delivery

   This approach integrated, tested, and deployed components when they are needed. It provides frequent opportunities for customer feedback loops.

3. Big Bang Deployment

   This approach deployed new or updated components to all targets simultaneously. It often required when dependencies prevent the coexistence of both old and new components. For example, a change in database schema may not be compatible with earlier versions of some components.

4. Pull Deployment

   In this approach, new or updated software is stored in a controlled repository, allowing users to download it to their devices at their convenience. It gives them control over when updates are applied.

Modern software development enterprises adopted devops practice, migrating the application workload to the cloud native deployment and refactoring monolithic applications to be a microservice-based architecture to increase agility (Batta, Shwartz, Nidd, Azad, & Kumar, 2021). It is largely influenced by Continuous Integration (CI), Continuous Delivery (CDE), and Continuous Deployment (CD). These practices streamline software release cycles by introducing automation, reducing manual effort, and improving software quality. Continuous Integration (CI) is a practice where developers frequently integrate code changes into a shared repository, often multiple times per day, and each integration undergoes an automated build and testing process to detect errors at an early stage. Building on this, Continuous Delivery (CDE) ensures that the software is always in a deployable state by automating all stages of the release process except the final deployment, which may still require human approval. Continuous Deployment (CD) takes this a step further by fully automating the final deployment, so that every build that passes the automated tests is automatically released into production without manual intervention (Shahin, Babar, & Zhu, 2017).

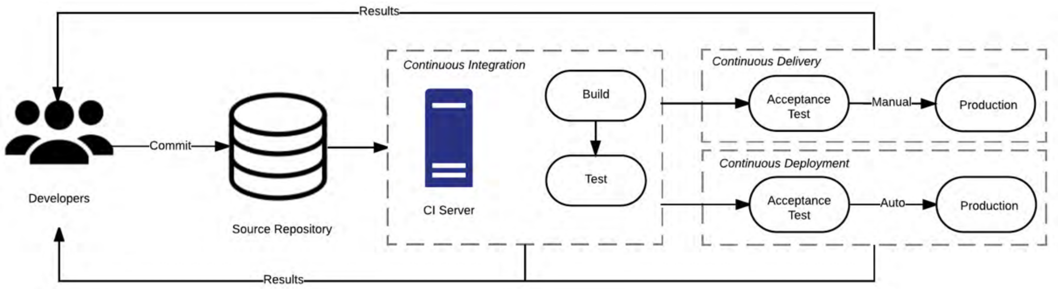


Figure II.2 The relationship between continuous integration, delivery and deployment (Prince, 2016)

Figure II.2 highlights the relationship between these three stages, showing that CI is a prerequisite for CDE, while CDE is a prerequisite for CD. Adopting CI/CD in deployment management offers multiple benefits. For instance, frequent deployments result in faster and more reliable releases, ensuring that updates and bug fixes reach users quickly. Automated testing and validation processes not only improve software quality by detecting issues early in the development cycle, but they also reduce deployment risks through continuous testing and incremental updates that minimize the potential for catastrophic failures. Additionally, CI/CD promotes collaboration between developers, testers, and operations teams by enabling them to work together in an automated pipeline. In the long run, this will lead to a better customer satisfaction and higher operational efficiency.

However, CI/CD implementation also comes with challenges. According to Shahin et al. (2017), many organizations struggle with infrastructure and tooling complexity. They often lack the necessary setup to fully automate testing and deployment. Another challenge is bottlenecks in testing which can occur when automated tests are slow or unreliable. It delays deployments and makes debugging more difficult. Cultural resistance is also a major issue, as teams who are used to traditional deployment methods may hesitate to adopt new automation practices. To successfully implement CI/CD, organizations need to invest in training, establish robust tooling, and shift toward a DevOps culture.

To address these challenges, organizations can follow best practices for deployment management in CI/CD environments. This includes automated testing to ensure all changes pass unit, integration, and acceptance tests before deployment. Infrastructure as Code (IaC), using tools like Terraform or Ansible, also helps automate infrastructure setup. It reduces human error and speed up deployments (Morris, 2016). Companies can also use progressive deployment strategies like blue-green deployments, which maintain parallel live and idle environments for quick rollback, or canary releases, which gradually roll out updates to a small group of users before full release. In addition to that, implementing real-time monitoring and observability tools can help detect deployment issues early (Kosińska, Baliś, Konieczny, Malawski, & Zieliński, 2023). By applying these strategies, companies can improve deployment reliability and reduce the risk of failures.

II.2 Prior Research

This section will examine previous studies that are related to this thesis. In 2016, a cloud outage study (COS) was conducted. It analyzed outages in 32 popular cloud services, such as WhatsApp, Twitter, and Youtube. A total of 597 outages data was collected from January 2009 to December 2015. Hardware upgrade/software update, network, and bugs were revealed as the top three root causes (Gunawi, et al., 2016). However, it didn't cover the solution or fix procedure to those problems. Simlarly, Aceto et al. from University of Napoli "Federico II" conducted a survey in 2018 about internet outages. The study analyzed data from various public sources and examine patterns and causes of internet outages across those datasets. It categorizes outages into hardware failures, routing issues, power disruptions, government censorship, and natural disasters (Aceto, Botta, Marchetta, Persico, & Pescape, 2018). However, their study primarily focused on IT infrastructure and industry-wide failures rather than internal enterprise IT incidents. This thesis, on the other hand, will not focus on the infrastructure-level disruptions. Instead, the focus will be more into application-layer failures and the organizational processes governing incident management.

In 2020, another related study was conducted about intelligent incident management. It analyzed 2 years data of incident tickets at Microsoft. This study identifies two key challenges in incident management within the context of large-scale cloud environments, namely the incomplete service/resource dependencies and the imprecise resource health assessment. The first challenge makes it difficult to pinpoint affected components or customer when incident happens, and it delays the root cause identification and impact estimation. The second challenge overwhelm engineers with redundant notifications that is caused by sensitive alerts. To overcome these challenges, the study introduces IcM BRAIN, an AI-driven incident management framework that leverage AI to infer relationships between incidents and resources and utilize machine learning techniques to improve the precision of anomaly detection and reduce alarm noise (Chen, et al., 2020). While the study was insightful and the solution (AI-enhanced monitoring) might detect problem earlier, its primary focus is still more toward the incident management, improving how to effectively mitigate an incident when it happens rather than preventing incidents from occurring in the first place. Furthermore, Microsoft, as a cloud service provider, operates at a different scale and with different failure patterns than a fintech company that provides direct consumer services. This thesis, in contrast, focuses on systemic failure patterns in software-defined businesses and explores preventive strategies for reducing incident occurrence rather than just improving post-incident handling.

In addition to studies on cloud outages and incident management, research has also explored the relationship between change management and incident prevention. Guven & Murthy (2016) examined the role of change in incident prevention, highlighting that 80% of IT incidents stem from system changes. Their study emphasized the importance of structured change reviews and predictive models to assess high-risk changes before deployment. More recently, Kapel (2023) further investigated incident prevention through reliable change deployments, proposing an approach to enhance deployment reliability by introducing a risk management AIOps framework that utilize real-world change, CI/CD pipeline, and incident data. While these studies focus on reducing failure risks in software deployments, they primarily emphasize pre-deployment risk assessment rather than analyzing post-mortem incident reports to identify failure patterns and systemic organizational weaknesses.

Table II.1 Summary Table of Previous Studies

Title	Methods	Key Contributions	Limitations
Why Does the Cloud Stop Computing? Lessons from Hundreds of Service Outages (2016)	Collected outage data from 32 cloud services using search engines with keywords: “serviceName outage month year”	• Introduce cloud outage study (COS) methodology. • Provided insights into broad availability issues in modern cloud services.	Did not analyze detailed root causes at an organizational level or explore specific incident prevention strategies.
A Comprehensive Survey on	Publicly available datasets, qualitative analysis	• Provided first large-scale survey on internet outages.	Focused on IT infrastructure-level disruptions, rather than application-layer

Internet Outages (2018)		• Identified root causes at different network layers (physical, data link, routing).	failures or incident management practices.
Towards Intelligent Incident Management: Why We Need It and How We Make It (2020)	Analysis of two years of Microsoft's incident tickets and postmortems, field studies on incident handling challenges.	• Identified key incident response challenges in large-scale cloud environments. • Proposed IcM BRAIN, an AI-based incident management framework for anomaly detection.	Focused on optimizing incident resolution, rather than incident prevention or systemic failure analysis.
Understanding the Role of Change in Incident Prevention (2016)	Analyzed IT change records and incident reports to extract explicit linkages.	Proposed predictive models to assess risky changes before deployment.	Focused on change risk assessment, but did not analyze incident detection and response inefficiencies.
Incident Prevention Through Reliable Changes Deployment (2023)	Conducts a single-case exploratory study by interviewing 15 subject matter experts at ING Bank	To provide insights into change and incident management and to develop a novel AIOps-based framework	This paper is a research plan rather than an empirical study, meaning it does not provide tested implementations or validated contributions yet.

Table II.1 presents a summary of the previous studies. To the best of author's knowledge, this research addresses the following gaps in prior studies:

1. A deeper, organization-specific analysis compared to previous surveys that examined outages at a broader global level. By focusing on a single fintech company, this study provides granular insights into how software-defined businesses experience and manage IT incidents.
2. A focus beyond change management, incorporating a systematic analysis of real-world post-mortem reports to identify not only failure patterns but also gaps in incident detection and response.

3. Proposes practical recommendations that address multiple aspects of incident management, including preventive measures, detection improvements, and response optimizations, making the insights applicable to similar fintech organizations.

II.3 Conceptual Framework

A conceptual framework is a description of how a researcher perceives the key factors or variables relevant to the study and the relationships between them. Its purpose is to define and connect the concepts being examined, using insights and evidence from existing literature (Rocco & Plakhotnik, 2009). The framework is designed based on prior studies on IT incident management, system failures, and response mechanisms, incorporating insights from various academic sources. It illustrates the key relationships between IT incidents, their root causes, incident management deficiencies, strategies for improvement, and their expected outcomes. Figure II.3 shows the conceptual framework of this research.

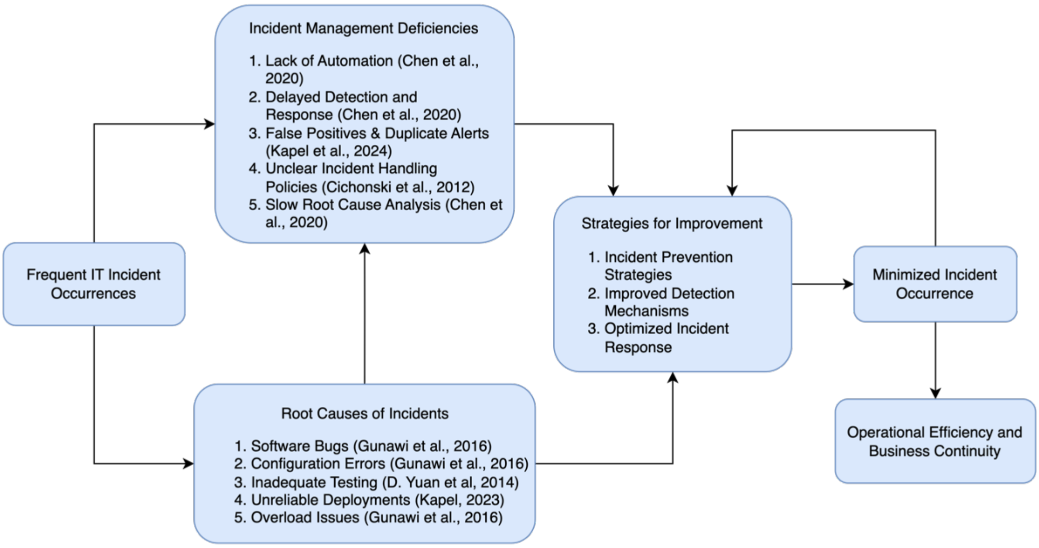

Incident Management Deficiencies
1. Lack of Automation (Chen et al., 2020)
2. Delayed Detection and Response (Chen et al., 2020)
3. False Positives & Duplicate Alerts (Kapel et al., 2024)
4. Unclear Incident Handling Policies (Cichonski et al., 2012)
5. Slow Root Cause Analysis (Chen et al., 2020)
Frequent IT Incident Occurrences
Strategies for Improvement
1. Incident Prevention Strategies
2. Improved Detection Mechanisms
3. Optimized Incident Response
Minimized Incident Occurrence
Root Causes of Incidents
1. Software Bugs (Gunawi et al., 2016)
2. Configuration Errors (Gunawi et al., 2016)
3. Inadequate Testing (D. Yuan et al, 2014)
4. Unreliable Deployments (Kapel, 2023)
5. Overload Issues (Gunawi et al., 2016)
Operational Efficiency and Business Continuity

Figure II.3 Conceptual Framework

The framework starts from the problem statement where frequent IT incidents needs to be addressed. They caused significant financial losses and potentially damage the company's reputation. The framework shows the root causes of IT incidents, such as software bugs, configuration errors, inadequate testing, unreliable deployments, and overload issues. Previous studies have shown that these factors contribute significantly to system failures.

Aside from root causes, the way incidents are detected and resolved also affects their impact. Several weaknesses in incident management can worsen system disruptions. One of the most common problems is false positives alerts. They produce a lot of notifications that cause alert fatigue for the oncall engineers. It can make critical issues being overlooked. Delayed detection and response also extend the duration of incidents, making the impact worse. The lack of automation also causes engineers to rely on manual processes. This increases the time required for detection and resolution.

To address these issues, the framework introduces strategies for improvement that focus on three key areas, namely incident prevention, improved detection mechanisms, and optimized incident response. The strategies will be based on the findings from root cause analysis and incident detection and response gaps analysis. Incident prevention strategies focus on improving software engineering processes. Improved detection mechanisms focus on identifying incidents as early as possible. Optimized incident response strategies focus on how the team can react quickly and correctly when an incident happens.

By implementing these strategies, the expected outcome is that the frequency and severity of IT incidents will be reduced. It ensures that disruptions become less frequent and easier to manage. Eventually, this will contribute to make business operational more efficient and improve business continuity. This conceptual framework becomes the foundation for this research in guiding the investigation of systemic failures in IT incident management at PT INUSA. With this framework, insights will be collected and then formulated to develop actionable recommendations for improvement.

Chapter III Research Methodology

Research methodology refers to the structured process of understanding and applying scientific methods to conduct research. It involves logically organized steps to address and solve research questions (Patel & Patel, 2019). This chapter describes the procedure to identify IT incident root causes, incident detection gaps, and incident response inefficiencies at PT INUSA. The chapter is divided into four sub-sections, which are research design, data collection method, data analysis method, and result validation.

III.1 Research Design

This section describes how the study is conducted. This research adopts a qualitative research design that focuses on understanding the primary causes of IT incidents and identifying detection and response gaps. This analysis will become the foundation to develop strategies for improvement. The study analyzes the post-mortem documents from PT INUSA. This study uses secondary data to answer the research questions. Figure III.1 shows the design of this research.

The process is started with Problem Identification. This stage explains about the main issue where frequent IT incidents are negatively impacting business performance. This has been discussed in detail in Chapter I. The next stage, Research Questions & Objectives Formulation, focuses on defining the goals of this research, which are to identify the primary causes of IT incidents, find the gaps in incident detection and response, and develop prevention and mitigation strategies. After this, a Literature Review is conducted to explore existing research, theories, and findings related to IT incidents, their detection mechanisms, and the response behaviors. In this stage, a comparative analysis of previous studies is also conducted to position this research contribution to the field of IT incident management.

The next stage, Data Collection, is an activity to select the appropriate data for this research and collect them. This research uses incident post-mortem documents at PT INUSA as the primary data source. The data will be collected through the internal company knowledge management system. These documents are chosen because they contain factual information and detailed post-incident analysis that is conducted by relevant teams shortly (at maximum few days) after incident is mitigated. This objectivity can help to reduce biases, especially when it is compared to interviews, where personal perspectives and memories may not be accurate. A post-mortem document also contains a rich historical context, such as incident timeline, root cause analysis, and lessons learned. These documents also provide insights on how incidents were detected, reported, and resolved.

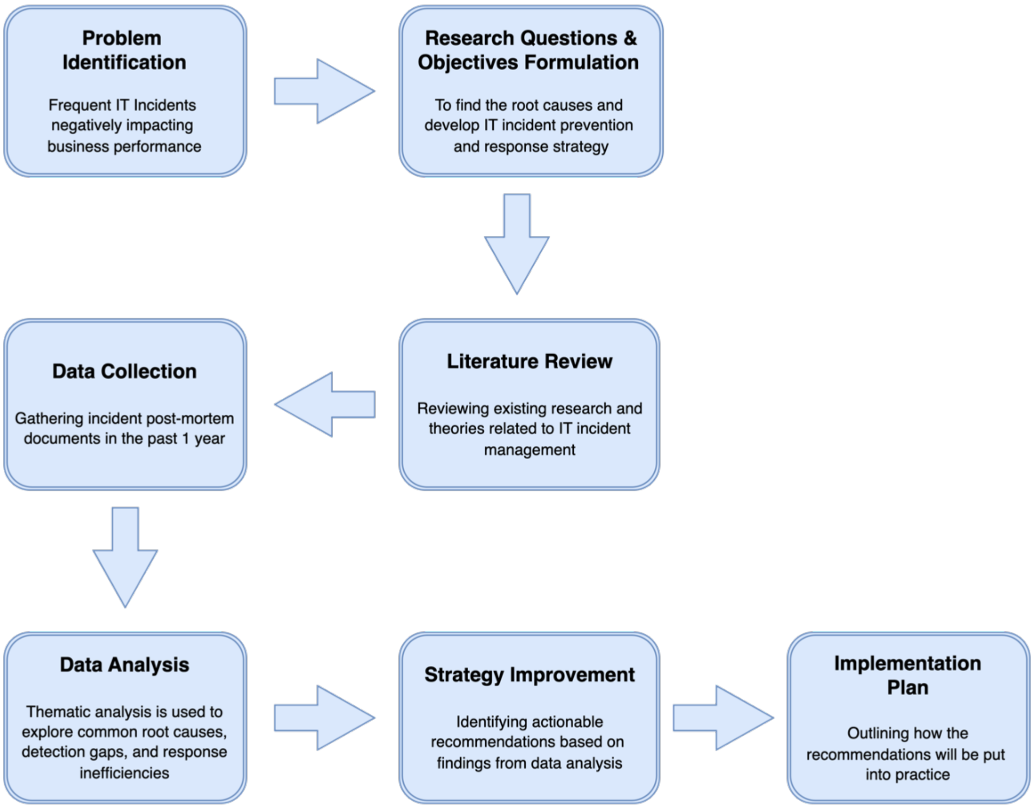

Problem Identification
Frequent IT Incidents negatively impacting business performance
Research Questions & Objectives Formulation
To find the root causes and develop IT incident prevention and response strategy
Literature Review
Reviewing existing research and theories related to IT incident management
Data Collection
Gathering incident post-mortem documents in the past 1 year
Data Analysis
Thematic analysis is used to explore common root causes, detection gaps, and response inefficiencies
Strategy Improvement
Identifying actionable recommendations based on findings from data analysis
Implementation Plan
Outlining how the recommendations will be put into practice

Figure III.1 Research Design

Once the data is collected, Data Analysis is then conducted using thematic analysis to explore common root causes, gaps in detection, and inefficiencies in incident response. This will be explained further in sub chapter III.3. The findings from this analysis will then be used in the Strategy Improvement stage to develop practical recommendations. These recommendations are expected to help companies improve incident prevention, detection, and response. Finally, the Implementation

Plan stage describes how these recommendations can be implemented by the company. It provides a project timeline, person in charge, and key milestones to help PT INUSA implement the proposed improvements into their existing business processes.

III.2 Data Collection Method

The primary data source for this study consists of 26 post-mortem documents from PT INUSA. It covers major incidents that occurred between August 2023 and August 2024. Each post-mortem document contains detailed information as described in Table III.1. The data was collected through the PT INUSA's internal knowledge management system where post-mortem documents are archived. The selection of documents focused on incidents which are attributed to four specific root cause categories: code, configuration, processes and policies, and testing. These categories were chosen based on their frequency, financial impact, and controllability as explained in the Sub-chapter I.3.

Table III.1 Details Provided by the Post-Mortem Documents

Item Name	Description
Incident Date	Represent the date when the incident started
Title	The incident title
Squad Owner	The team who owns the incident and post-mortem document. This is decided based on the cause that triggers the incident.
GMV Impact	The estimation of total GMV (opportunity) loss
Revenue Impact	The estimation of total Revenue (opportunity) loss
Time Incident Started	The date time when incident began
Time Incident Detected	The date time when incident is detected whether it is by an automated alert or manually by an engineer / employee or based on complaints from users.
Time Incident Resolved	The date time when incident ended
Time Post-mortem Closed	The date time when the post-mortem document is complete and all corrective action items are implemented.
Root Cause Category	The attributed root cause category of the incident agreed by all participants during the post-mortem review session.
Incident Summary	The summary of the incident that briefly explain about what, when, and why it happened, and how incident was mitigated. This typically consisted of one or two paraghraps.

Item Name	**Description**
Impact	Description of what functionality was disrupted during incident and the scale of it, e.g., ten thousand of users located in certain area were unable to make transactions in the platform.
Trigger	Event or action that triggers the incident.
Detection	Description on how the team detected the incident.
Root Cause Analysis (Using 5 Whys)	Detailed information and root cause analysis of the incident. The analysis is conducted using 5 Whys method.
Timeline	Describes events that happened and actions that were taken during incidents. These events and actions are ordered chronologically.
Resolution and Recovery	The list of actions that were taken to resolve the incident.
Corrective and Preventive Measure	The list of action items from the incidents. There are 3 types of action items: corrections, preventions, and improvements.
Lessons Learned	The list of learnings from the incident explaining about what went well, what went wrong, and what luck that fortunately help the team to prevent further impact.
Related Squads	All the teams that are impacted by the incident. An incident can impact many services that are owned by multiple teams.

Author believes that the selected data sources are suitable to answer research questions. The first research question is to find the primary causes of incidents. The post-mortem document contains root cause analysis which is highly relevant for identifying patterns across incidents. Other data that can also be used to find the causes are the incident triggers and incident timeline. Using these data, important insight can be collected such as root cause classifications, frequency of incidents by each category, and then patterns that trigger incidents.

The second research question focuses on identifying factors that delay incident detection and response. The post-mortem documents provide relevant data for this, such as how incident was detected and when it is started and detected. This information can be used to analyze gaps in incident detection. Another data, incident timeline, will also be useful to analyze inefficiencies in response behavior from the team that handle the incident. By analyzing these data, this research can identify gaps in incident detection and response that contribute to longer incident duration.

The third research question aims to propose strategies to improve overall IT incident prevention and management processes. The insights obtained from the first two research questions will become the foundation to develop these strategies. The proposed strategies will then be compared and supported by industry best practices from the literature review to help validate the proposed strategies.

To conclude this section, there is a limitation of using post-mortem documents. Even though they contain a lot of important and relevant information, their completeness depends on the thoroughness of the engineering teams who wrote the documents. There also might be some aspects that were not captured in the post-mortem documents. To mitigate potential biases because of these, this study focuses on identifying patterns across multiple incidents rather than only rely on individual cases. This approach can help to ensure that findings are based on recurring patterns or systemic issues rather than isolated events.

III.3 Data Analysis Method

This section will outline the data analysis methods that will be used to analyze the post-mortem documents at PT INUSA. The post-mortem documents contain many relevant data for this research, such as root cause analysis, lessons learned, the explanation about the trigger of the incident and how an incident is detected, and many more. The complete data can be seen in Table III.1. Based on these data, the following methods will be used to do the analysis:

1. Tagging (Coding) Process

   Tagging (also referred to as coding) is a method that is used to make sense of text data by indexing or mapping segments of the data to provide an overview of different concepts. This process enables researchers to understand the data in relation to the research questions (Elliott, 2018). Tagging is a basic step in thematic analysis where segments of the text are labeled with tags that represent specific concepts. This process is used to organize the data and help to make it easier for identification of themes. Tagging the data systematically will enable author to break down the complex information into manageable parts that can make it easier to identify patterns and their relationships.

In this study, an open-source tool called Taguette is used to help with the tagging process. It ensures consistency in doing categorization of incident-related information. To avoid confusion between “code” in qualitative analysis and “code” in software engineering, the term “tagging” is used instead of “coding”. Each tag will only have a maximum of single occurrence per post-mortem document even though it is mentioned multiple times in it. This is done to make sure that a factor in a single incident is not over-represented.

2. Thematic Analysis

Thematic analysis is a widely used qualitative data analysis method that can be used to identify and analyze patterns (themes) within a data set. An important feature of thematic analysis is the active construction of themes. It is the process where researchers are getting involved with the data to identify themes that emerge from the data set. This is an active process where researchers must be critical to analyze and also interpret the data rather than only summarize it (Kiger & Varpio, 2020).

This study uses the six-phase thematic analysis framework by Braun & Clarke (2006) that is already adopted and accepted by many researchers. The first phase is familiarization with the data. In this case, it means that the researcher reviews all post-mortem documents to understand the incident context. This step enables the researcher to have an initial understanding of the data before doing a detailed analysis. After that, the step is to generate initial tags by identifying failure patterns, detection gaps, and response inefficiencies in the incidents. These tags will become the foundation for further thematic exploration.

After the tagging process is completed, the next phase is to search for themes. It is an activity where related failure patterns are grouped together to form a broader concept. The primary activity of this phase is to identify recurring trends in IT incidents rather than isolated cases. After themes are identified, they will be reviewed and refined again to make sure that the theme accurately represents the data. Those themes are then given the name to classify incident root causes and incident management gaps. Finally, the last phase will produce the final analysis, where the themes are compiled and processed to provide insights that will be used to improve overall incident management processes.

Thematic analysis is considered sufficient and appropriate for identifying root causes in this study because the data source (post-mortem documents) contains factual and structured information based on real incidents. Each post-mortem includes a detailed root cause analysis (using the 5 Whys method), triggers, incident timeline, and lessons learned. These documents are created shortly after incidents occur which ensures the relevance and reliability of the evidence. Given the textual and evidence-based nature of post-mortems, this method provides a rigorous foundation to uncover underlying causes and gaps in incident management.

Tagging and thematic analysis will be applied sequentially. First, tagging is conducted to identify patterns related to incident root causes and gaps in incident handling. After that, they will be categorized to form the themes that represent a broader concept. With this approach, researcher can understand the systemic issues comprehensively. It provides insights not only for incident prevention but also for improving incident detection and response processes.

III.4 Result Validation

To ensure the credibility of the research findings, a validation step will be conducted after the thematic analysis is completed. The goal of this validation is to confirm whether the identified root causes and recommended strategies align with actual conditions and challenges faced by the organization.

The validation will be performed by presenting the research findings to the senior engineering leader at PT INUSA. As someone who oversees the engineering division and has a deep experience in incident management and overall engineering processes, the senior leader is in a strong position to assess the accuracy, relevance, and feasibility of the proposed recommendations. Their input will help verify whether the findings reflect systemic issues and provide valuable insights for developing the improvement strategies.

This validation step is important to ensure that the outcomes of this study are practically applicable. Insights from the leader may also be used to refine the recommendations and adjust the implementation plan in a way that better fits the organization's current priorities and constraints.

Chapter IV Results and Dicsussion

This chapter presents the findings of the research based on an analysis of 26 post-mortem reports of major IT incidents. It is organized into four main sections. The first section discusses about the root causes of IT incidents. The second section explores gaps in incident detection and response. The third section provides recommendations to address these root causes. Lastly, the fourth section presents the implementation plan of the proposed solutions so that it can be implemented by PT INUSA.

IV.1 Incident Root Causes

This section analyzes the root causes of incidents based on insights from 26 post-mortem reports. The findings are first categorized and later consolidated in the final section of this subchapter. Each category is examined in detail to identify patterns and recurring issues that contribute to incidents. This analysis serves as the foundation for the business solutions and implementation plans presented in the following sections.

IV.1.1 Code-Related Root Causes

Code-related issues were the most frequently assigned root cause category where they account for 12 of the 26 major incidents. As explained in Chapter 3, thematic analysis is used to uncover incident root causes from post-mortem documents. The first step of thematic analysis is tagging. From 12 post-mortem documents in this category, a total of 36 tags related to contributing factors of incidents were generated. The number of unique tags from those 36 is 26. Table IV.1 shows the list of tags with its occurrences.

Table IV.1 Code-Related Post-mortem Tags

Tag	Number of Occurrences
Lack of Staging Tests	5
Incomplete Validation Logic	2
Unhandled/Mishandled Error in Code	2

Tag	Number of Occurrences
Insufficient Change Review Process	2
Deployment Process Deficiency	2
Incomplete Test Coverage	2
Invalid Query Logic	2
Misconfigured Cache Mechanism	1
Missing Feature in Migration	1
Missing Configuration	1
Improper Load Management	1
Logic Misunderstanding	1
Missing Requirements	1
Missed Code Implementation	1
Timeout Misinterpretation	1
Missing Database Index	1
Incomplete Regression Test Coverage	1
Infrequent Deployment	1
Dependency Between Base App and Feature Module	1
Unfamiliarity of Service Deployment	1
Lack of Linter or Code Checker	1
Unimplemented Callback Handling	1
Unlisted Change in the Deployment Document	1
Unoptimized Code	1
Accidental Deployment	1
Local cache usage	1

The number of tag occurrences is unique per incident. It means, when the tag has more than 1 occurrence, then that factor contributes to multiple incidents, indicating a recurring issue. The description of each tag and the complete list of occurrences can be seen in Appendix B and Appendix D, respectively. To provide a clearer understanding of the patterns, the tags were grouped into broader themes. These themes were developed by identifying shared characteristics among the tags, focusing on the underlying processes or systems involved. Table IV.2 summarizes the themes, their associated tags, and explanations for why they were grouped together.

Table IV.2 Code-Related Post-mortems' Themes

Theme	Tags	Explanation
Testing and Validation Gaps	Lack of Staging Tests, Incomplete Validation Logic, Incomplete Test Coverage, Incomplete Regression Test Coverage, Missing Requirements.	These tags indicate weaknesses in the testing process. These gaps increase the likelihood of undetected errors that included into production environment.
Deployment and Process Deficiencies	Insufficient Change Review Process, Deployment Process Deficiency, Unlisted Change in the Deployment Document, Unfamiliarity of Service Deployment, Infrequent Deployment	These tags highlight process gaps in change reviews and deployment practices. Poorly documented changes and inadequate deployment processes contribute to errors during code integration and rollout or release.
Code Logic and Implementation Issues	Invalid Query Logic, Mishandled Error in Code, Missed Code Implementation, Logic Misunderstanding, Missing Feature in Migration, Unoptimized Code, Missing Database Index, Local Cache Usage, Dependency Between Base App and Feature Module, Timeout Misinterpretation, Unimplemented Callback Handling	These tags shows direct implementation errors, such as incorrect logic, missed features, or inefficient code. Even though these causes are significant, these issues are often unique to specific incidents rather than systemic.
Configuration and Infrastructure Issues	Misconfigured Cache Mechanism, Missing Configuration, Improper Load Management	These tags relate to the technical environment in which the code operates. Misconfigurations or infrastructure gaps often make the impact of code-related issues worse.
Tooling and Automation Gaps	Lack of Linter or Code Checker, Insufficient Change Review Process, Unlisted Change in the Deployment Document	These tags indicate missing tools or automation that could have prevented errors.

Based on Table IV.2 combined with the reference to Table IV.1, the code-related root causes (code logic and implementation issues) turned out to be mostly unique for each incident. Among the 26 unique tags identified, only a few were recurring. Specifically, "Invalid Query Logic," "Mishandled Error in Code," and "Incomplete Validation Logic" appeared in more than one incident, each of them occurring twice. Interestingly, despite being categorized as code-related incidents, the most frequent root cause identified was related to testing issues, followed by deficiencies in deployment processes and change review practices. This suggests that while code implementation errors are significant, systemic gaps in testing and deployment processes play a larger role in enabling such incidents to occur. However, these aspects will be discussed further in the next sections.

These are the findings for each theme from 12 post-mortem documents with code category:

1. Testing and Validation Gaps

   Testing-related issues were the most frequent theme, with "Lack of Staging Tests" occurring in five incidents. For example, in the incident #2024080704 (See Appendix C), insufficient staging tests allowed untested scenarios to cause disruptions in top-up feature. Similarly, "Incomplete Validation Logic" contributed to two incidents where business rules were not properly implemented in the code. These findings indicate a need for more comprehensive test coverage. Testing-related root causes will be discussed further in Section IV.1.4.
2. Deployment and Process Deficiencies

   "Insufficient Change Review Process" and "Deployment Process Deficiency" each appeared in two incidents, highlighting the risks of unreviewed or poor deployment execution. For instance, the accidental deployment of previously reverted commits (Incident #2023101701) underscores the need for better rollback mechanisms and deployment safeguards to prevent unwanted changes to be released. This will be further discussed in section IV.1.3.

3. Code Logic and Implementation Issues

   Code logic issues were primarily unique to specific incidents. For example, "Invalid Query Logic" caused inefficiencies in database operations, while "Missed Code Implementation" led to incomplete features. These findings suggest the importance of stricter code reviews and adherence to coding standards. This will be further discussed in this section.
4. Configuration and Infrastructure Issues

   Even though this theme is less frequent, configuration-related problems, such as "Misconfigured Cache Mechanism" and "Missing Database Index," had significant impacts on system performance. These issues indicate a need for better infrastructure monitoring and configuration management. This will be discussed further in Section IV.1.2.
5. Tooling and Automation Gaps

   The absence of tools like linters contributed to issues such as runtime application server crash due to the codes that make ereference to an unexist variable (Incident #2023101901). Automating code checking processes could prevent similar incidents in the future.

Code logic and implementation issues, while often unique to individual incidents, reveal systemic gaps in code reviews and adherence to development standards. These gaps can lead to critical issues that impact system reliability and user experience. One such incident, #2024042201, provides valuable insights into how insufficient approval during the code review process allowed a minor change to cause a major disruption. The following case highlights the risks of inadequate code reviews.

Case Study: Incident #2024042201 – Some Users Are Getting Stuck on a Particular Page

Incident Overview:

On April 22, 2024, a critical incident occurred in the PT INUSA’s app where users attempting to access the QRIS feature were stuck on an unresponsive page. This disruption affected a quite significant portion of the app's user base, resulting in complaints and a decline in user satisfaction.

Root Cause Analysis:

The root cause was traced to a change in the build.gradle file, where an unnecessary dependency was added without proper review during the code review process. This dependency caused conflicts during runtime that leads to the unresponsive page. The change had been approved in the code review process by the team who handle QRIS feature. However, there was no approval from the core apps team who is responsible for managing the `gradle` file.

Organizational Context:

The code review process at PT INUSA is typically conducted within the same squad responsible for the feature being modified. In this case, the team handling QRIS feature approved the change without consulting the core apps team, which manages the gradle file and its dependencies. PT INUSA uses GitLab for version control, where code reviews are conducted through merge requests. However, the process lacked safeguards to ensure that changes affecting shared or core components, such as the gradle file, were reviewed and approved by relevant teams outside the feature squad. This lack of cross-team collaboration and ownership contributed to the approval of the unnecessary dependency.

Lessons Learned:

This incident shows the need for several improvements in PT INUSA’s code review and approval processes:

1. Equip engineers and reviewers with better knowledge of the implications of changes to configuration files, such as the gradle file. Training programs or documentation can help engineers understand the potential risks of adding unnecessary dependencies.
2. Use tools like GitLab CODEOWNER to enforce approval workflows for specific parts of the codebase. This would ensure that changes to critical files, such as the gradle file, require explicit approval from designated individuals or teams (e.g., the core apps team) before being merged into the main branch. Without such approval, the changes cannot proceed.

From the case study, it is evident that a well-structured code review process can play a crucial role in preventing incidents. Returning to the focus of this section (code-related root causes of incidents), many of the identified code logic issues were unique to specific incidents rather than systemic. Nevertheless, they provide valuable lessons that can be applied not only within PT INUSA but also in other organizations. To summarize and conclude this section, the following table (Table IV.3) presents the key findings from the analysis of code logic and implementation issues.

Table IV.3 Code-Related Root Causes

Tag	Explanation
Incomplete Validation Logic	Changes in the cronjob logic to extend budget period accidentally created duplicate promo budget rows for the same period. This issue arose because the logic failed to validate whether the period was already covered, leading to inconsistent promo budget data.
Invalid Query Logic	In one incident, the query lacked validation for the active budget period, resulting in the system retrieving incorrect rows. In another case, a query was duplicating wallets when activating the new payment method feature. It leads to unregistered wallet retrievals and user confusion.
Local Cache Usage	The use of local cache instead of centralized cache, combined with a code bug significantly increase traffic to the authentication server where it continuously ask for new tokens everytime it makes a request. This made the auth server unable to handle the load.
Logic Misunderstanding	A developer misunderstood the early return logic in the code, believing that a top-up feature would always be disabled for blacklisted users. This misunderstanding led to incorrect system behavior, demonstrating the importance of clear documentation and code readability.
Missing Database Index	The absence of an index on critical in the transactions table caused query inefficiencies, leading to slower system performance. Adding the missing index resolved the performance bottleneck.
Missing Feature in Migration	A logic/feature for handling "Request In Progress" status was omitted during the migration to the new system, despite being present in the legacy system. This omission caused gaps in processing ongoing requests, leading to operational inconsistencies.

Tag	Explanation
Timeout Misinterpretation	The system incorrectly treated a timeout as a failed transaction instead of a pending one, triggering retries through alternative channels. This caused double transfers, one from Channel A and another from Channel B, resulting in financial losses.
Unhandled/Mishandled Error in Code	A broken validation check in the system allowed the flow to continue to the third party banking server, resulting in another timeout and compounding transaction issues. This highlights the need for stricter error-handling mechanisms to prevent cascading failures.
Unimplemented Callback Handling	The system failed to implement callback handling in the Payment 2.0 system. It makes billing statuses pending even after receiving a successful callback from the third party partner. This gap caused disruptions in payment status updates and delayed transaction processing.
Unoptimized Code	A slow query in the auto-refund process caused delays during high transaction volumes. The query verified transaction eligibility for refunds but lacked optimization, resulting in prolonged refund processing times when multiple transactions were stuck.

IV.1.2 Configuration-Related Root Causes

Configuration-related issues were the second most frequent root cause category, accounting for 7 of the 26 major incidents. As with code-related root causes, thematic analysis is used to identify contributing factors from the post-mortem documents. Through the tagging process, a total of 23 tags were generated for configuration-related incidents. Fourteen of them were unique. These tags highlight systemic challenges such as deficiencies in change execution processes, unclear ownership, and missing configurations. There are also gaps in monitoring and migration strategies. Table IV.4 presents the tags and their occurrences. Same as before, the number of occurrences is unique per incident. The description of each tag can also be seen in Appendix B.

Table IV.4 Configuration-Related Post-mortem Tags

Tag	Number of Occurrences
Change Execution Process Deficiency	3
Lack of Migration Strategy Guideline	3

Tag	Number of Occurrences
Missing Configuration	3
Deployment Process Deficiency	2
Lack of Monitoring Activity	2
Unclear Ownership and Communication	2
Manual Certificate Renewal	1
Configuration Management Deficiency	1
Lack of Staging Tests	1
Knowledge Transfer and Handover Deficiency	1
Configuration Error	1
Incomplete Monitoring	1
Change Approval Bottleneck	1
Insufficient Change Review Process	1

Table IV.5 Configuration-Related Post-mortems' Themes

Theme	Tags	Explanation
Change Management and Execution Issues	Change Execution Process Deficiency, Deployment Process Deficiency, Change Approval Bottleneck, Insufficient Change Review Process	These tags highlight deficiencies in managing and executing configuration changes. Poorly planned changes, delayed approvals, and insufficient reviews often result in undetected errors. Similar to code-related issues, these problems emphasize the need for robust change management practices.
Configuration and Migration Challenges	Missing Configuration, Configuration Error, Configuration Management Deficiency, Lack of Migration Strategy Guideline	These tags show challenges with system configurations and migrations. Missing or incorrect configurations, together with the absence of migration strategy guidelines, result in inconsistencies during transitions. This is similar to code logic issues where missing features caused failures.
Monitoring and Communication Gaps	Lack of Monitoring Activity, Incomplete Monitoring, Unclear Ownership and Communication	As in code-related incidents, monitoring and communication gaps are recurring themes. Incomplete monitoring leads to delayed detection, while unclear ownership and poor communication slow down issue resolution.
Knowledge and Process Deficiencies	Knowledge Transfer and Handover Deficiency, Manual Certificate Renewal,	These tags highlight deficiencies in knowledge transfer and reliance on manual processes, such as certificate renewal. Similar to testing gaps in code-related incidents, limited staging

	Lack of Staging Tests	environments and inadequate handovers increase the likelihood of errors.

Based on Table IV.4, the configuration-related root causes turned out to have a mix of recurring and unique contributing factors. Among the 14 tags identified, "Change Execution Process Deficiency," "Lack of Migration Strategy Guideline," and "Missing Configuration" were the most frequent where each of them occurring in three separate incidents. This highlights recurring challenges in managing configuration changes and executing migrations effectively. Furthermore, tags such as "Deployment Process Deficiency" and "Unclear Ownership and Communication" show systemic issues in communication and process management, which worsen the impact of configuration-related problems. These tags are then grouped into themes explained in Table IV.5. These are the findings for each theme from the seven post-mortem documents with configuration category:

1. Change Management and Execution Issues

   "Change Execution Process Deficiency" being the most frequent tag in this theme, occurring in three incidents. For example, in the incident #2023092102 (see Appendix C), a poorly planned change to the configuration caused downtime in the system. Similarly, "Deployment Process Deficiency" appeared in two incidents, highlighting risks in poorly managed deployment executions. These findings indicate a need for stronger change management practices, such as detailed planning and review processes.

2. Configuration and Migration Challenges

   Tags such as "Missing Configuration" and "Lack of Migration Strategy Guideline" appeared frequently, with each occurring in three incidents. For instance, in #2024020601, a missing configuration during a flash sale caused an error in stock availability that makes users unable to purchase. In #2023081001, the lack of a migration strategy caused an important activity to be skipped during the cutover. As a result, many requests were unable to be served by the server.

3. Monitoring and Communication Gaps

Monitoring and communication-related issues, such as "Lack of Monitoring Activity" and "Unclear Ownership and Communication," were also shown in several incidents. Each of them occurred in two incidents. In #2024031903, incomplete monitoring activity caused a gateway error in the e-money system. It was not detected until users reported the problem.

4. Knowledge and Process Deficiencies

   Knowledge-related gaps, such as "Knowledge Transfer and Handover Deficiency" was also shown. For example, in #2024020601, an incomplete knowledge transfer process during team handovers caused an incident undetected for quite a long time. The automated alerts were actually already in-place. However, the team forgot to update the receiver of the alert so the alerts were never received by the right team.

Configuration-related issues are often getting worse due to systemic challenges, such as inadequate migration strategies and poor deployment process. One incident, #2024022901, illustrates how missing configurations during a data migration in a critical service caused widespread disruptions. The following case highlights the risks of missing configurations and the absence of detailed migration planning:

Case Study: Incident #2024022901 – Stuck Transaction in Paid State Increase for Phone Credit Products

Incident Overview:

On February 29, 2024, and again on March 1, 2024, incidents occurred where a significant number of phone credit transactions became stuck in the "Paid" state without completing. This prevented users from receiving their purchased credits and data plans. It caused substantial operational disruptions and financial losses.

Root Cause Analysis:

The primary root cause was a missing Redis key migration during the transition to a shared Redis instance. This migration was part of a cost-saving initiative, but one of Redis keys, which is critical for transaction processing, was not migrated. As a result, the expected feature was disabled, and it processed the transaction using the old flow that is already deprecated. It caused a significant lag and transactions become stuck in the "Paid" state.

Organizational Context:
PT INUSA had a company wide Redis migration project. It is a project to merge Redis instances to become a single instance to save cost. However, the taskforce that led this project did not provide a standardized migration strategy. As a result, each team executed the migration independently. In this case, the team who is responsible for the migration, manually review the source code to find Redis keys that need to be migrated to the new instance. However, during the cutover, issue happens and after some investigation, it is found that a critical Redis key was not migrated. It was not listed in manual review assessment to be migrated. This mistake won't happen if there is a standardized strategy, e.g., to create a script that automatically migrate critical Redis keys. One of the criteria for the critical key is the key that has no time-to-live (TTL) configured. It indicates that the key is always there and might be used.

Lessons Learned:
This incident shows the importance of having standardized processes to manage configuration changes, especially for cross-team initiatives like this:

1. Taskforces who drive the projects must create clear guidelines. For Redis migrations, reusable scripts should be developed to migrate all keys, especially the one without TTL.
2. Automated tools should be implemented to validate the completeness of configurations during migrations. This tool can check the discrepancies between old and new environments to ensure no critical data are missed.

From the case study above, it is shown that missing configurations can directly trigger major incidents. Going back to the overall configuration-related issues, there are some recurring problems that directly caused incidents. They are missing configurations, misconfigurations, and reliance on manual processes. These issues highlight the importance of proper execution and validation during configuration changes. To summarize and conclude this section, Table IV.6 presents the key findings from configuration-related issues that directly triggered incidents.

Table IV.6 Configuration-Related Root Causes

Tag	Explanation
Configuration Error	This tag highlights errors or human mistake during configuration setup. For example, a production configuration was misconfigured in the centralized configuration repository using a staging value.
Lack of Staging Tests	This tag identifies gaps in pre-deployment testing. For example, one of the services used rate limit threshold that were not comparable to production traffic. It leads to inaccurate testing and eventual overload in the production environment.
Manual Certificate Renewal	Reliance on manual processes during SSL certificate renewal caused delays and disruptions. For example, the team renewed the certificate but failed to update the Kubernetes cluster, as the initial task did not include it in the instructions.
Missing Configuration	A recurring issue in multiple incidents where critical configurations were overlooked or omitted. Examples: missing critical Redis key during migration and incomplete routing configuration for Kafka during the cutover to shared Kafka.
Unclear Ownership and Communication	This tag refers to the lack of clarity in roles and responsibilities during configuration changes. For instance, in the case of one of third party IP address configuration change, no one clarified with the partner on which IP should be used. It leads to indecision and delays until the old IP was unable to be accessed.

IV.1.3 Process-Related Root Causes

Process and policy-related issues accounted for 4 of the 26 major incidents analyzed. As with the other root cause categories, thematic analysis was used to identify the factors contributing to these incidents from the post-mortem documents. Through the tagging process, a total of 11 tags were generated, with several recurring across multiple incidents. These tags highlight gaps in deployment processes, approval workflows, inter-team communication, and policy adherence. Those gaps directly triggered significant disruptions. Table IV.7 presents the tags and their occurrences. It provides a detailed overview of the process and policy-related root causes. The tag description can be seen in Appendix B.

Table IV.7 Process-Related Post-mortems' Tags

Tag	Number of Occurrences
Deployment Process Deficiency	2
Insufficient Change Review Process	2

Tag	Number of Occurrences
Unclear Ownership and Communication	2
Limited Pre-deployment Check	1
Miscommunication Partner Notification	1
Accidental Deployment	1
Lack of Monitoring Activity	1
Lack of Staging Tests	1

From Table IV.7, it is evident that process-related root causes are diverse. They consist of deficiencies in deployment and pre-deployment processes, insufficient reviews, and communication gaps. The most recurring issues in this category were "Deployment Process Deficiency," "Insufficient Change Review Process," and "Unclear Ownership and Communication," each occurring in two incidents. Other tags, such as "Limited Pre-deployment Check" and "Accidental Deployment," highlight the consequences of inadequate processes and policies that allowed errors to reach production. These findings emphasize the importance of structured workflows, clear communication, and thorough reviews to prevent process-related incidents. To summarize and provide deeper insights, Table IV.8 groups the tags into broader themes.

Table IV.8 Process-Related Post-mortems' Themes

Theme	Tags	Explanation
Change Management and Execution Issues	Deployment Process Deficiency, Limited Pre-deployment Check, Accidental Deployment	This reflects insufficient safeguards in managing deployments. These tags highlight issues such as missing logs, unintentional deployments, and inadequate pre-deployment validation. For example, accidental deployment of a canceled commit caused one of the partner's transactions to fail.
Review and Approval Gaps	Insufficient Change Review Process	This theme points to the lack of rigorous reviews and approvals for changes impacting services or products. For example, the one of the banks' Kirim Uang configuration change was executed without necessary approvals from the product owner squad.

Monitoring and Communication Gaps	Lack of Monitoring Activity, Unclear Ownership and Communication, Miscommunication Partner Notification	These tags highlight weaknesses in detecting and addressing issues early due to inadequate monitoring and unclear communication. For example, one of partners' notifications were sent to the wrong team that cause a delay to resolve.
Testing and Validation Gaps	Lack of Staging Tests	This tag highlights insufficient testing environments for validating changes. For example, in the incident that involves one of game voucher's payments, the lack of proper staging tests for immediate invoice expiration resulted in unhandled scenarios in the payment process.

Based on Table IV.7 and Table IV.8, process and policy-related root causes are driven by deficiencies in IT change management, review and approval workflows, testing environments, and communication processes. Among these, the most recurring themes are Change Management and Execution Issues, Review and Approval Gaps, and Monitoring and Communication Gaps. Below are the key findings for each theme:

1. Change Management and Execution Issues

   Issues related to IT change management and execution were among the most frequent process-related root causes, with "Deployment Process Deficiency," "Limited Pre-deployment Check," and "Accidental Deployment" contributing to multiple incidents. For instance, in the incident #2024021901, a canceled commit from a previous deployment was accidentally included in a subsequent deployment due to incomplete deployment logs and the absence of a clear flag that indicates the commit had been canceled. Similarly, in the incident #2023112401, a change to the Kubernetes HPA API resulted in incorrect configuration, causing service disruptions. These findings highlight the importance of robust deployment processes. They should include comprehensive change logs and automated safeguards to prevent unintentional changes from reaching production.
2. Review and Approval Gaps

   Tags such as "Insufficient Change Review Process" indicates gaps in the review and approval workflows. They were identified as contributing factors in two incidents. For example, in the Kirim Uang incident (#2023112401), a change about a product-specific configuration was executed without approval from the product owner squad. In incident #2023110301, the fraud team did not notify and ask for approval from the payment team about the new rule. As a result, when it is deployed, invoice got expired even though user balance is already deducted.

3. Monitoring and Communication Gaps

   Monitoring and communication-related deficiencies also contributed to multiple incidents. They are represented by the tags "Unclear Ownership and Communication" and "Miscommunication Partner Notification." In the BPJS incident (#2023102601), the partner's notification about an IP address change was sent to the wrong internal team. The notification was not forwared to the right team, so it caused system integration failure. In another incident, poor communication between the Fraud and Payment teams caused an issue in the Game voucher payment flow.

4. Testing and Validation Gaps

   Insufficient testing also contributed to process-related incidents. In the Game Voucher payments incident (#2023110301), the lack of comprehensive staging tests caused an untested critical change to be included to the production release. This highlights the importance of creating thorough pre-production testing environments to identify and mitigate potential failures before deployment.

From the findings discussed above, it is evident that process and policy-related root causes often coming from deficiencies in change management, communication, and review workflows. These gaps not only disrupt operational efficiency but also create opportunities for critical errors, as seen in multiple incidents. One illustrative example is the Phone credit incident (#2024021901) where an accidental deployment caused widespread disruptions. The following case study delves deeper into this incident to highlight the impact of flawed deployment processes and the lessons it provides.

Case Study: Incident #2024021901 – No Pulsa and Paket Data Transactions to Partner A

Incident Overview:

On February 19, 2024, an issue occurred where no pulsa and paket data transactions were successfully processed through the Partner A. This disruption began on February 15 and persisted until February 19, resulting a significant opportunity loss. Customers who attempted to make transactions experienced automatic refunds due to backend processing errors. The issue was detected late by the Partner A team and escalated to PT INUSA.

Root Cause Analysis:

The root cause of this incident was an accidental deployment of the Partner A API migration change. This change had been part of a previously canceled deployment but was mistakenly included in the subsequent release. The issue arose because the deployment logs did not clearly indicate that the earlier deployment had been canceled. It also did not flag that the commit is reverted. As a result, the deployer assumed the changes were ready for production and deployed them without proper validation. The deployment process lacked safeguards. There is no automated system to identify and exclude canceled changes which caused this oversight to occur. There was also no pre-deployment check to confirm that all included changes were intentional.

Organizational Context:

At PT INUSA, even though the deployment pipeline was developed by DevOps team for all the product engineering teams, the deployment process itself is managed by individual squads. There is no centralized oversight for this. The Partner v2 API migration was expected to improve system integration, but it must be released when both PT INUSA and the partner are ready. However, in this case, due to no standardized deployment log format, the changes are not tracked consistently. Furthermore, pre-deployment reviews are manual and rely on the deployer's awareness of previous changes. This increases the risk of human error.

Lessons Learned:

This incident showed weaknesses of PT INUSA's deployment processes. These are some of the aspects to be considered for future improvements:

1. Standardized Deployment Logs
   Implementing a standardized format for deployment logs can help to make the changes to be tracked consistently. It should mark the canceled changes as well.
2. Automated Safeguards
   Developing tools to automate the validation of deployment change log would reduce the risk of accidental deployments. It would be better if the tool can exclude unintended changes.
3. Enhanced Pre-deployment Checks
   Developing a pre-deployment checklist or approval process would improve validation. This checklist should ensure only intentional changes are released by requiring all change authors to acknowledge their change for the deployment.

The above incident illustrates the impact of weaknesses in the deployment process. It can cause accidental deployments and also delayed incident detection. Addressing these gaps is important to prevent similar incidents in the future and improve the reliability of the deployment workflow.

IV.1.4 Testing-Related Root Causes

Testing-related issues accounted for 3 of the 26 major incidents analyzed. Through the tagging process, a total of 12 tags were generated, with several recurring across multiple incidents. These tags highlight gaps in testing coverage and regression testing which directly caused service disruptions. Table IV.9 presents the tags and their occurrences. It provides a detailed overview of testing-related root causes. The description of the tag can be seen in Appendix B.

Table IV.9 Testing-Related Post-mortems' Tags

Tag	Number of Occurrences
Lack of Knowledge	2
Lack of Staging Tests	2
Incomplete Regression Test Coverage	2
Incomplete Test Coverage	2
Slave DB Lag	1
Knowledge Transfer and Handover Deficiency	1
Idempotency Check Issue	1
Invalid Query Logic	1

Table IV.9 shows that testing-related root causes often come from knowledge gaps and incomplete testing practices in the staging environment. There are some recurring issues such as "Lack of Staging Tests," "Incomplete Regression Test Coverage," and "Incomplete Test Coverage". They indicate systemic weaknesses in the testing process. To better understand these findings, the tags have been grouped into broader themes, as shown in Table IV.10.

Table IV.10 Testing-Related Post-mortems' Themes

Theme	Tags	Explanation
Testing and Validation Gaps	Lack of Staging Tests, Incomplete Regression Test Coverage, Incomplete Test Coverage	These tags highlight weaknesses in testing coverage, especially in staging. For instance, in the gaming incident, the lack of automation and staging tests resulted in unhandled scenarios for transactions without a reference_id.
Knowledge and Process Deficiencies	Lack of Knowledge, Knowledge Transfer and Handover Deficiency	These tags indicate gaps in knowledge sharing and testing behavior. For example, the logic error in the Gaming API came from a limited understanding of query behavior and lack of validation.
Data Integrity and Validation Gaps	Slave DB Lag, Idempotency Check Issue	These tags illustrate issues about data consistency and integrity. For example, the Gaming API uses slave database that has a potential of lagging behind master. This situation caused redundant transactions during high traffic.
Code Logic and Implementation Issues	Invalid Query Logic	This tag highlights the impact of defect in the code logic. For instance, using JOIN instead of LEFT JOIN in the Gaming API query caused transactions without a partner_reference_id to fail. It then triggered unnecessary refunds.

Table IV.10 shows that the most common issue in testing-related incidents is the lack of comprehensive test coverage in the staging environment. Recurring tags such as "Lack of Staging Tests," "Incomplete Regression Test Coverage," and "Incomplete Test Coverage" indicate systemic deficiencies in testing processes that fail to handle edge cases. These gaps increase the risk of incidents. In addition to that, the recurring theme of "Knowledge and Process Deficiencies" shows that insufficient knowledge transfer can cause major incidents, such as the Android SDK upgrade incident. To understand more about the real-world impact from these testing deficiencies, the following case study discusses one of the most severe incidents at PT INUSA caused by lack of testing.

Case Study: Incident #2024042101 – Logic Error in Gaming Order Status API

Incident Overview:

On April 18, 2024, a logic error in the Gaming Supply Service API caused discrepancies between transaction statuses in Gaming Supply Service and Gaming Demand Service. Transactions without a reference_id were mistakenly flagged as not found (TID_NOT_FOUND). This condition triggered auto-refund mechanism in the Demand Service. This error caused a significant financial loss of around IDR 467.6 million because the user's balance was refunded but the transaction has been processed.

Root Cause Analysis:

The root cause of this incident was a logic error that was introduced in the new deployment. The API query used JOIN instead of LEFT JOIN, causing transactions without a reference_id to be excluded. The patch was deployed without proper test case coverage for scenarios that involves missing reference_id values. Furthermore, the testing environment lacked automated regression tests to validate the changes. There were no staging tests to simulate this real-world transaction flows.

Organizational Context

At PT INUSA, testing processes were previously managed by dedicated test engineers who owned the responsibility for designing and executing tests, including maintaining automation frameworks. However, in mid-2023, a company-wide initiative, referred to as the "Test Revamp" project, restructured the testing approach. As part of this initiative, the number of test engineers was gradually reduced. Some test engineers were converted to become software engineers, but the majority of them were let go.

After this change, the responsibility for testing was given to software engineers. This restructure aimed to integrate testing with development workflows. However, it introduced challenges. Because of no dedicated roles, many automated tests have become outdated or are no longer well-maintained. This shift in responsibilities contributed to gaps in testing processes.

Lessons Learned:

This incident provided several lessons to improve testing practices:

1. Test cases should be developed comprehensively to cover all possible scenarios.
2. Regression tests should be automated and integrated to the deployment pipeline to validate changes before deployment.

This incident becomes a reminder that addressing these testing deficiencies through automation will help PT INUSA minimize the risk of similar incidents in the future.

IV.1.5 Summary of Incident Root Causes

The analysis in the previous sections revealed that 80% of major incidents were triggered by internal changes, such as new feature deployments, system migrations, and configuration modifications. This indicates that failures mainly come from software development, deployment, and operational processes rather than external factors like infrastructure failures or unexpected traffic surges. Several recurring systemic issues contributed to incidents across multiple root cause categories. Figure IV.1 summarizes the most common failure patterns identified in the analysis.

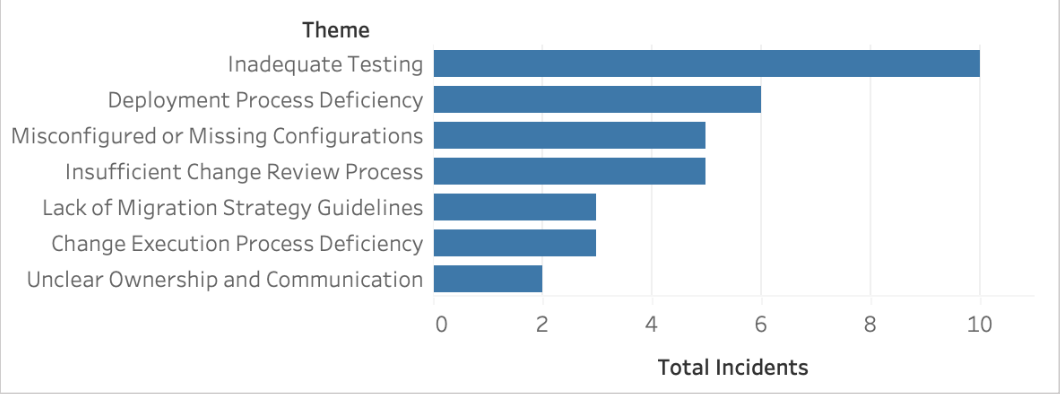


Figure IV.1 Systemic issues contributing to IT incidents

The most frequently observed issue was inadequate testing, contributing to 10 out of 26 incidents. Many deployments lacked regression testing, leading to failures in existing critical functionality. Additionally, incomplete test coverage resulted in undetected failures that only surfaced in production environments. Deployment deficiencies were the second most frequent cause, accounting for 6 incidents. In some cases, manual deployment change logs contained errors or outdated information, leading to unreviewed changes being applied. Additionally, a lack of automated deployment validation resulted in production environments missing essential configurations.

Insufficient change review processes led to multiple incidents where modifications were merged or applied without proper peer or product owner approvals. Similarly, misconfigured or missing configurations caused system failures, as engineers often configured settings in staging but forgot to apply them to production. Change execution deficiencies and lack of standardized migration strategies were also notable contributors to incidents. Teams often failed to actively monitor key metrics after changes, leading to delays in identifying failures. Furthermore, during infrastructure migration initiatives, due to large amount of work, each team executed their migrations independently with no guidance from the taskforce team who drive the project, leading to knowledge gaps and unexpected failures. Lastly, ownership ambiguities and communication failures also contributed to several disruptions where third-party IP changes were not communicated effectively between internal teams, leading to integration failures.

IV.2 Incident Detection and Response Gaps

As discussed in Chapter I, PT INUSA's Mean Time to Detect (MTTD) of 82.5 hours and Mean Time to Resolve (MTTR) of 38.5 hours indicate that there is a significant room for improvement. These delays not only prolong the impact of incidents but also increase operational costs and negatively affect user experience. This section examines the underlying causes of poor MTTD and MTTR by analyzing PT INUSA’s post-mortem documents, especially in the Detection and Timeline section.

At PT INUSA, incidents are detected and handled through a combination of automated systems, user reports, and escalations from external channels such as social media complaints that are handled by customer service agents. Figure IV.2

describes the incident handling process. Once an incident is identified, it is reported to the product owner. Product owner then will report the incident in a dedicated channel and inform the Incident Manager. For major incidents, the Incident Manager escalates the issue to a war room, which often involves the Core Team (including software architects) and the IT Infrastructure Team. This collaboration aims to expedite root cause identification and resolution. In cases involving third-party systems, the Product Engineering Team coordinates with the external party to address the issue. Simultaneously, admins are tasked with ensuring that relevant announcements are communicated on the platform to keep users informed. After resolution, a post-mortem document is created to analyze the incident and document the lessons learned for future improvements.

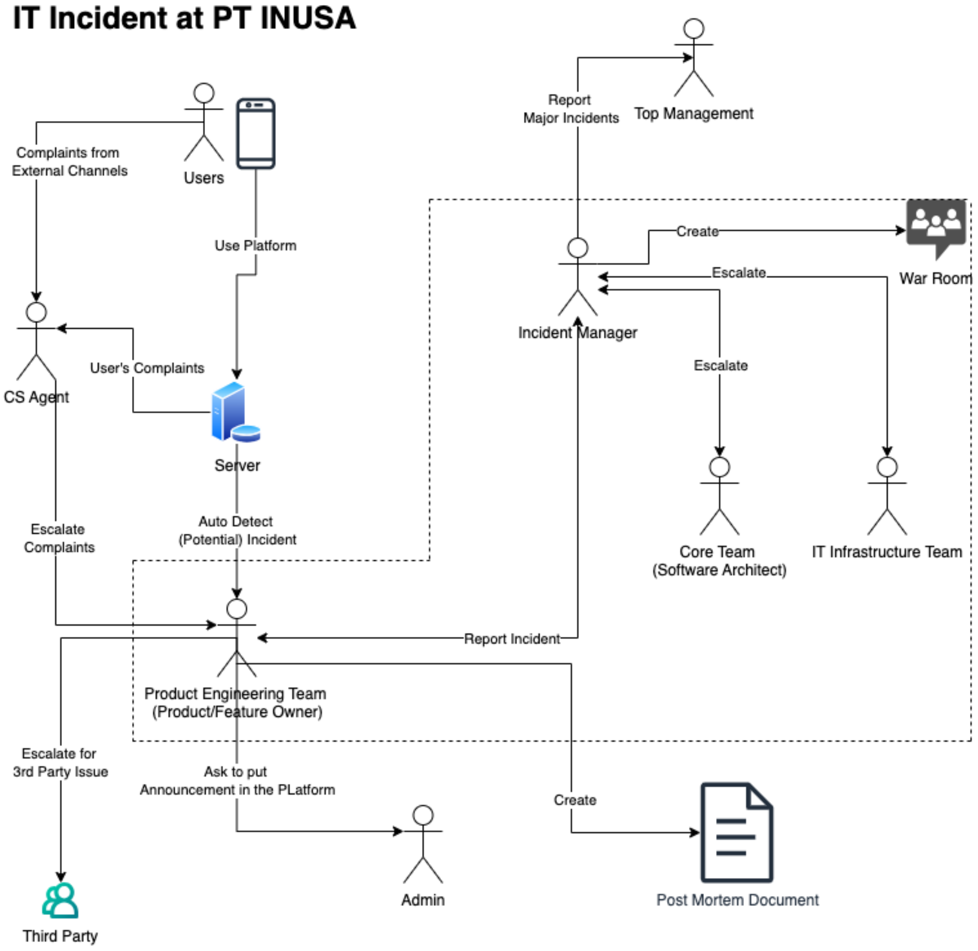


Figure IV.2 Incident Handling at PT INUSA

To further analyze the underlying causes of poor Mean Time to Detect (MTTD), a breakdown of incidents tagged with "Missing Alerts" was conducted. Eighteen incidents out of 26 (69%) were tagged with "Missing Alerts". These incidents highlight gaps in the current monitoring and alerting systems at PT INUSA. Table IV.11 summarizes the specific content related to missing alerts and categorizes them into relevant alert types. It ranges from transaction drops to app crashes. This analysis provides valuable insights into areas where detection mechanisms can be improved to reduce delays in identifying and addressing incidents.

Table IV.11 Breakdown of Missing Alerts Tags

Incident Number	Content indicating Missing Alerts	Alert Category
2024080501	Complaint Increase, especially from Kirim Uang Users ... Actions: create alert for any unknown status received (expand to the other vendors as wel)	Third Party Unknown Transaction Status
2024070301	Detection: Double mutation report by the finance team. The reconciliation process is manual y sent to Finance, thus our system doesn't have visibility to create double mutation monitoring	Double Mutation
2024070202	Auth Service CPU was impacted due to the increase in load, but had no CPU alerting making it harder to investigate	High CPU Utilization
2024042201	Detection: Complaint from users … Some users are getting stuck on a page in the app when trying to open the QRIS feature	App Crash / Stuck
2024013001	Detection: User complains through our CS, and CSM reporting to us through slack … User cannot create Product A transaction in all Platform	Transaction Drop for Specific Product
2024011801	Detection: We had a few user complaints … Team business sudah pernah raise di akhir desember dari metrics ada penurunan topup dari apps akan tetapi bersamaan dengan failure topup API dari sisi Third Party di tanggal 13 des 2023 dan resolusi selesai di tanggal 6 januari 2024	Transaction Drop for Specific Product
2023101702	Detection: Product team first detected the issue when getting complain from user … Impact: Opportunity lost to use Payment Method B as payment method	Transaction Drop for Specific Payment Method

Incident Number	Content indicating Missing Alerts	Alert Category
2023101701	Detection: There is a report in support channel that said there are several products from Partner A that were closed.	Transaction Drop for Specific Partner
2023081501	Detection: Raise from payment product team … Impact: Product B transaction with Payment Method A dropped to zero for 5 days.	Transaction Drop for Specific Payment Method
2024070102	Detection: Complaint from traders … Impact: Customers can't transact Electricity Prepaid, Property tax, and Vehicle Tax products that are supplied by Partner B	Transaction Drop for Specific Partner
2024031903	Detection: The PM noticed an unexpected metric when he monitored the e-money product metrics … Impact: Users can't make e-money transactions for almost 5 days	Transaction Drop for Specific Product
2024020601	Detection: There is a heads up from Employee A regarding the GMV metrics is decreased since 1 Feb … Product C users are unable to do check out when pressing bayar, users wil get error out of stock	Transaction Drop for Specific Product
2024021901	Detection: The Partner C team noticed that there has been no successful pulsa and paket data transaction from PT INUSA since February 16 and they notified our team	Transaction Drop for Specific Partner
2023110301	Detection: Report from CSM team that there are raising complaints about several Payment methods on #payment-support	Transaction Drop for Specific Payment Method
2023102601	Detection: Business team request to change the IP for Partner D … Impact: GMV loss from 25 October 2023 at 16.00 until 26 October 2023 at 10.24 for BPJS product	Transaction Drop for Specific Partner
2024071101	Detection: Got a report from the product team and business team … Impact: User unable to open app from push notification	Visit Drop from Specific Channel

Incident Number	Content indicating Missing Alerts	Alert Category
	sent by Third Party B. Tapping it do nothing, does not redirect to the respective screen in the app	
2024052901	Detection: We got some report from our user that they cant do QRIS registration … App v2.28 wil crash when user do QRIS registration.	App Crash / Stuck, User Registration Drop
2024042101	We are lucky that a partner raised this issue to Employee B. Then the employee relay the info to the team … Impact: Financial loss due to successful transaction in gaming supply service side but it's being refunded in gaming demand service side.	Refund Rate Increase

From the analysis, several patterns emerge. Transaction-related drops, such as "Transaction Drop for Specific Product," "Transaction Drop for Specific Payment Method," and "Transaction Drop for Specific Partner," were the most frequently recurring alert categories. These incidents often relied on manual detection methods, such as complaints from users or reports from the product team. It indicates an overdependence on reactive monitoring rather than proactive alert systems. For example, in Incident #2024011801, the team only became aware of the issue after multiple user complaints and a business team report highlighted a drop in the top-up activity.

Another common gap was in detecting app crashes or system errors, such as "App Crash/Stuck" incidents in #2024052901 and #2024042201. These incidents were identified through user complaints rather than automated monitoring tools. Similarly, incidents like #2024020601 and #2023101702 revealed a lack of alerting for metrics drops, such as GMV decreases or product-specific checkout errors, which could have been detected earlier with well-configured alerts. Moreover, the lack of alerts for specific technical issues, such as high CPU utilization (#2024070202) or unknown transaction statuses from third-party systems

(#2024080501), indicates that the current alerting mechanisms are not comprehensive enough to cover all critical scenarios. This shows gaps in both the breadth of monitoring configurations and the depth of their implementation.

While improving detection mechanisms is crucial, another significant challenge lies in PT INUSA’s incident response practices. An analysis of the company’s top 9 incidents with the time-to-resolution of more than 3 hours (summarized in Table IV.12) reveals that there are several recurring issues that delay incident resolution. One prominent finding is delayed incident reporting. There are 4 out of 9 incidents where issues detected through user complaints or monitoring tools were not promptly escalated to the appropriate channels or teams even though reporting procedures are already in place. For example, in one incident (#2024042201), a critical anomaly was detected on April 18, but it took four days before it was formally reported as an incident. This shows some behavioral factors that cause delay in resolution. Engineers used informal communication channels and giving the impression of no sense of urgency.

In addition to delayed reporting, the analysis uncovered inefficiencies in immediate action and coordination after an incident is escalated. For example, in the loan wallet duplication incident (#2023101702), there was a significant delay in deploying fixes even though the root cause has been found early. In another case, dependencies on external partners in incident #2024080501, introduce additional layers of complexity. It makes the timely resolution become more challenging.

Lastly, the lack of post-fix monitoring was also observed as a systemic issue. It is shown in the QRIS registration screen crash incident (#2024052901). Even after an initial fix was deployed, the same issue reoccurred because key performance metrics were not actively monitored to verify the fix result. This situation highlights the need for a comprehensive post-resolution monitoring process to ensure the incident is fully mitigated and does not reemerge. Table IV.12 provides a detailed summary of the incidents and their tags related to incident response.

Table IV.12 Incident Response Issues Tags

Incident Number	Tag	Explanation
2024052901	Lack of Post-Fix Monitoring	After deploying the initial fix in version 2.28.1 (on the same day after the issue is detected), the team did not actively monitor crash metrics to verify its effectiveness. The recurring crash was eventually flagged by another engineer 11 days after the 50% rollout began. If proper monitoring processes had been in place, the recurring issue could have been detected and addressed earlier.
2024042201	Delayed Incident Reporting	The incident was detected on April 18, 2024, but the incident report was only created on April 22, 2024, leading to a delay of four days. This delayed reporting prolonged the resolution process and reflects a gap in the escalation procedure.
2024011801	Delayed Resolution Due to Lack of Immediate Action	The anomaly was reported on January 12, 2024, but the bug was only identified on January 16, 2024, after a four-day delay The timeline highlights a significant delay in identifying the root cause after the issue was detected. This suggests inefficiencies in the debugging or escalation process that could have prolonged the incident resolution
2024080501	Delayed Resolution Due to Dependency on External Partner	The resolution process was delayed because it relied on adjustments from the external partner. Although the team acted promptly by regrouping and escalating the issue, the dependency on partner's system changes prolonged the resolution process
2023101702	Delayed Resolution Due to Lack of Immediate Action, Delayed Incident Reporting	While a war room was created, there was a delay in deploying the fix, which could have been expedited. The time gap between identifying the root cause and deploying the fix suggests room for improvement in streamlining resolution workflows --- There was a two-hour delay in escalating the issue from the support channel to the incident support channel. This delayed the initiation of resolution efforts
2024071101	Delayed Incident Reporting, Prolonged Testing and Deployment Process	The issue was detected at 5:17 PM on July 11, 2024, but it was only reported to the incident channel at 8:53 PM --- The fix was merged at 10:32 AM on July 12, 2024, but the app hotfix was only released on Google Play Store at 4:08 PM, almost six hours later

Incident Number	Tag	Explanation
2024070301	Slow Incident Resolution	The issue was reported by the finance team on July 3, 2024, at 10:55 AM, but the fix was only deployed and tested by 5:05 PM on the same day, over six hours after detection -- While the team acted promptly by toggling off the auto-retry mechanism and specific channels early in the investigation, the overall resolution took several hours. This delay may indicate inefficiencies in the debugging, fix development, or deployment process
2024021901	Delayed Incident Reporting	The issue was reported in the support channel at 9:14 AM, but it was only reported as an incident at 1:18 PM, nearly four hours later
2024020601	Delayed Resolution Due to Late Escalation	The anomaly affecting Product A transactions was found at 09:00 AM on February 5, 2024, but the deployment to resolve the cron issue did not occur until 3:00 PM on the same day

The analysis of post-mortem reports reveals two major areas of inefficiencies in PT INUSA's incident management process: incident detection gaps and incident response inefficiencies. These gaps highlight the need for systemic improvements in both monitoring configurations and incident handling workflows. Table IV.13 provides a summary of these gaps, categorizing the key issues and their description. Addressing these deficiencies will be essential for reducing downtime and improving overall operational resilience.

Table IV.13 Summary of Incident Detection and Response Gaps at PT INUSA

Identified Gaps	Description
Missing alerts for key metrics	69% of incidents lacked proper alerting, e.g., transaction drops, CPU spikes (#2024070202, #2024013001). This gap increased Mean Time to Detect (MTTD) as incidents detected via manual reporting instead of automated alerts
Delayed incident reporting	Some incidents were escalated hours or even days after initial detection (#2024042201, #2024071101)
Slow root cause identification and debugging	Some fixes were delayed despite early detection of root cause (#2023101702, #2024011801)
Lack of post-fix monitoring	Issues resurfaced due to insufficient verification after resolution (#2024052901)

IV.3 Strategies to Improve Incident Prevention, Detection, and Response

The analysis of incidents in the previous sections has revealed several recurring issues and systemic gaps that contribute to major incidents at PT INUSA. Addressing these issues requires targeted solutions that not only resolve the root causes but also enhance the company's overall resilience in incident prevention, detection, and response. This subchapter proposes actionable solutions to mitigate the identified root causes and improve incident management processes. The solutions are categorized into two main areas, root cause mitigation and improvements in incident detection and response.

Table IV.14 Mapping of Identified Root Causes and Their Solutions

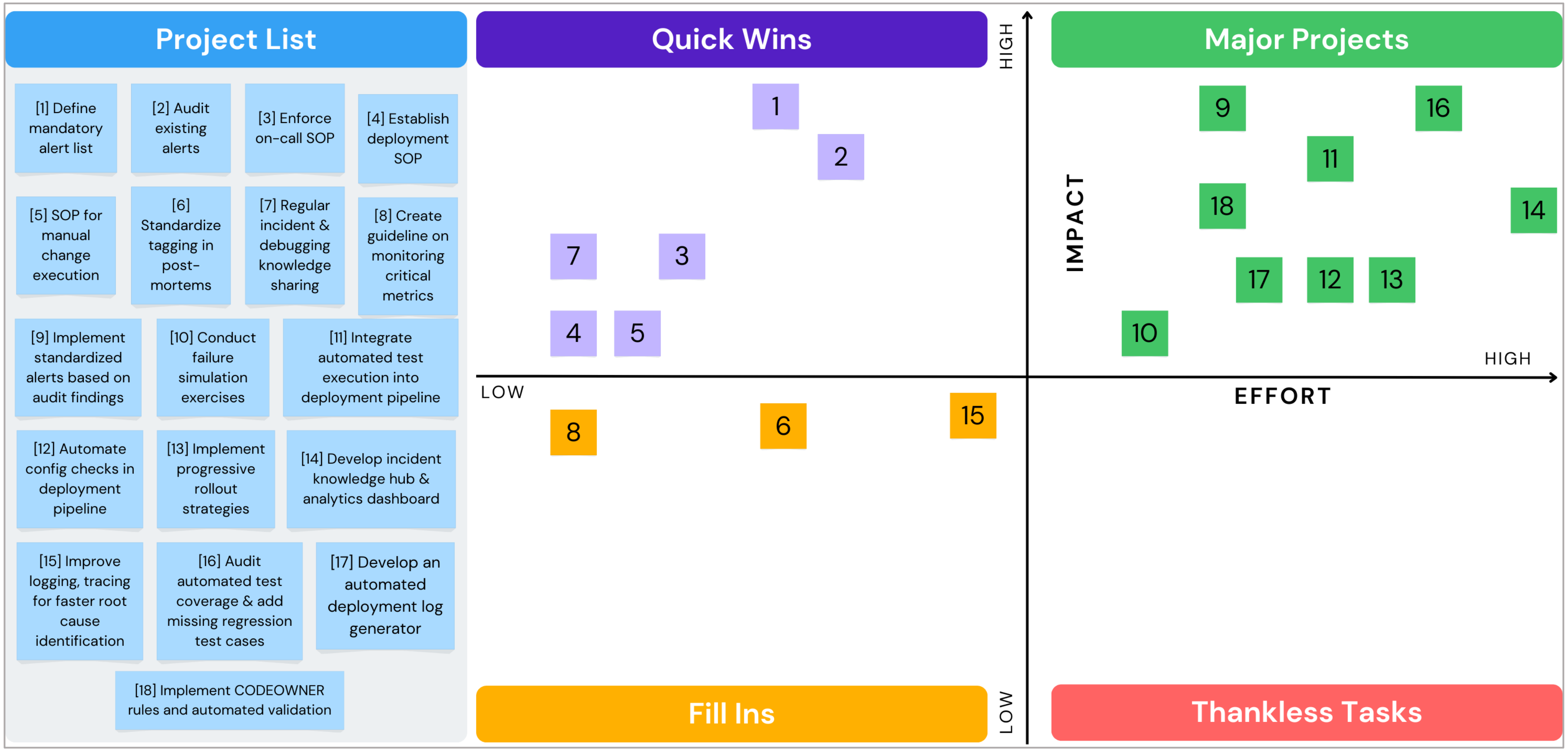
Root Causes	Solutions
Incident Primary Causes	
Inadequate Testing	Implement automated regression testing and enforce test coverage thresholds
Deployment Process Deficiency	Adopt progressive rollout strategies (e.g., canary releases), strengthen deployment validation, and automate deployment change log
Misconfigured or Missing Configurations	Enforce pre-deployment validation by integrating configuration checking into the deployment pipeline
Insufficient Change Review Process	Enforce change approvals by relevant teams using CODEOWNER feature
Lack of Migration Strategy Guidelines	Develop standardized migration playbooks
Change Execution Process Deficiency	Define clear execution procedures and enforce monitoring during change rollouts
Unclear Ownership and Communication	Introduce clear procedure to receive critical notification from third party to ensure the notification goes to the right team.
Incident Detection & Response Gaps	
Missing alerts for key metrics	Introduce alert standardization and audit existing alerts based on the standard.
Delayed incident reporting	Enforce mandatory escalation procedures and establish centralized incident tracking
Slow root cause identification and debugging	Improve logging, tracing, for faster root cause identification; conduct a regular failure simulation exercise; conduct a regular incident and debugging knowledge sharing
Lack of post-fix monitoring	Enforce monitoring and sanity testing after the fix is implemented. Create a guideline on how to monitor critical metrics and logs.

Table IV.14 provides a mapping of the identified root causes and the corresponding solutions. These solutions will be further elaborated in the following sections. However, while the table provides a structured mapping of root causes and corresponding solutions, not all solutions can be implemented simultaneously due to limited resources. To effectively allocate resources, an Action Priority Matrix is used to categorize solutions based on their effort required and expected impact on improving incident prevention, detection, and response. Figure IV.3 illustrates this matrix. It highlights Quick Wins that should be implemented immediately, Major Projects that require substantial resources, and Fill-Ins that are beneficial but lower in priority.

IV.3.1 Solutions to Address Incident Root Causes

Addressing the root causes of incidents at PT INUSA requires a comprehensive approach that targets testing deficiencies, deployment and change management processes, configuration and infrastructure gaps, and communication challenges. The following solutions are proposed to mitigate these root causes and build a more reliable system for incident prevention.

One of the primary themes identified in the analysis was testing process deficiencies. It includes gaps in staging environments, regression test coverage, and overall test case completeness. To address this, PT INUSA should prioritize the re-establishment of ownership over testing responsibilities. Although testing is currently integrated into the software engineer’s role, this has led to inconsistent coverage and neglected automation maintenance. PT INUSA could consider reintroducing a dedicated testing role or creating a cross-functional team specializing in testing and quality assurance. If there is a budget limitation, an alternative approach would be to mandate all teams to regularly conduct a thorough audit of existing automated test scenarios (in a quarterly basis). This audit should aim to identify gaps in coverage, especially related to critical business flows.



Project List

- [1] Define mandatory alert list
- [2] Audit existing alerts
- [3] Enforce on-call SOP
- [4] Establish deployment SOP
- [5] SOP for manual change execution
- [6] Standardize tagging in post-mortems
- [7] Regular incident & debugging knowledge sharing
- [8] Create guideline on monitoring critical metrics
- [9] Implement standardized alerts based on audit findings
- [10] Conduct failure simulation exercises
- [11] Integrate automated test execution into deployment pipeline
- [12] Automate config checks in deployment pipeline
- [13] Implement progressive rollout strategies
- [14] Develop incident knowledge hub & analytics dashboard
- [15] Improve logging, tracing for faster root cause identification
- [16] Audit automated test coverage & add missing regression test cases
- [17] Develop an automated deployment log generator
- [18] Implement CODEOWNER rules and automated validation

Figure IV.3 Action Priority Matrix

Subsequently, efforts should be made to enhance and expand the automated test suite to ensure comprehensive coverage of these critical areas. There also should be an SOP to ensure that newly developed features are covered in the automated test suite. Integrating these automated tests into the deployment pipeline is also essential. Deployments should be configured to proceed to production only if all automated tests pass successfully. This practice aligns with ITIL's emphasis on Service Validation and Testing, which ensures that deployed releases and the resulting services meet customer expectations and verify that IT operations can support the new service (Lawless, ITIL Service Validation And Testing, 2024).

Another critical area to improve is the deployment and change review process. Several incidents have shown that there are issues coming from inadequate code reviews, improper configuration changes, and deployment execution gaps. To mitigate these risks, PT INUSA should enhance its Merge Request (MR) review process. The proposed solution is to implement linters and GitLab’s CODEOWNER feature to enforce code quality standards and ensure that Merge Requests cannot be merged without appropriate approvals from relevant teams. Other than that, an auto configuration validation is also recommended to be applied in the MR pipeline to prevent misconfigurations such as inputting staging values in the production host config. This can be prevented by a simple regex-based validation to check common staging or preproduction address patterns. It can reduce human errors that caused incidents.

Beyond MR reviews, deployment pipeline automation should be strengthened. Automated testing, as discussed in the previous section, should be integrated into the pipeline before any production deployment to act as a gatekeeper for release quality. Furthermore, a configuration verification step should be added, which compares production configurations against staging to identify missing or incorrect settings before deployment occurs. A notable improvement would be the automatic generation of deployment logs using a standardized template. This log should include a complete list of changes that will be applied, which must be auto-generated by comparing the upcoming release tag with the current live version.

Each listed change must automatically tag the engineers responsible for it to ensure clear visibility into what modifications are being introduced. This structured logging mechanism would prevent issues caused by incomplete or inaccurate change documentation, a problem that has occurred in past incidents.

To further enhance deployment reliability, progressive rollout strategies should be reintroduced. In PT INUSA's legacy infrastructure, progressive rollout was achieved through canary deployment. It allows incremental rollouts. However, in the new infrastructure, this capability is not yet available, increasing the risk of widespread impact from faulty deployments. Restoring progressive rollout would enable PT INUSA to gradually release changes in controlled stages, starting with a small percentage of users or requests, then increasing progressively based on monitoring results. This approach allows teams to detect and address issues early, before they escalate to the entire user base. If anomalies are detected, the rollout can be paused or rolled back immediately to prevent widespread disruptions. Progressive rollouts are widely used by industry leaders such as Google and Facebook, which leverage them to ensure stability and minimize risks during deployments (Beyer, Jones, Petoff, & Murphy, 2016). By reintroducing this capability, PT INUSA can reduce deployment-related incidents, enhance deployment flexibility, and improve overall system stability.

To reinforce accountability and execution discipline, a revised Deployment Standard Operating Procedure (SOP) should be established. Under this SOP:

- All engineers involved in a change must acknowledge their modifications in the deployment log (e.g., Confluence document) before deployment begins.
- Executors must obtain explicit approval from a senior engineer or an engineering manager before initiating deployment.
- Deployments should not proceed if acknowledgments from all change authors and approvals from managers are incomplete.

Finally, a structured SOP for change execution should be created and it must incorporate post-release monitoring and sanity testing as required steps. Several past incidents could have been prevented or mitigated earlier if active monitoring and validation had been conducted immediately after the change was applied. This SOP should also cover approval workflows, stakeholder announcements, and contingency plans in case rollback is necessary. These improvements align with ITIL's Change Management & Deployment Management principles, which emphasize controlled, tested, and approved deployments to minimize operational risks (Lawless, ITIL Deployment Management, 2023).

Beyond technical improvements, addressing gaps in knowledge sharing and communication is essential to prevent recurring issues. While PT INUSA already has a blameless postmortem culture, ensuring that lessons learned are distributed across teams remains a challenge. To improve accessibility and application of postmortem insights, PT INUSA should enhance its incident knowledge hub. Although postmortems are already stored in a dedicated Confluence space, their effectiveness can be improved by implementing standardized tagging for better searchability and automated insights extraction via dashboards. These enhancements would enable engineers to quickly identify recurring patterns, contributing factors, and impacted services so they can avoid repeated mistakes.

Additionally, regular incident sharing sessions should be introduced to ensure that critical insights reach all relevant teams. Not all engineers proactively review postmortems, and valuable lessons are often only beneficial to the directly affected squads. By establishing a bi-weekly or monthly incident review meeting, PT INUSA can create a structured platform where incident owners present recent incidents, key takeaways, and systemic gaps identified. These discussions should go beyond technical fixes to explore preventative strategies that can be applied across teams.

IV.3.2 Solutions for Incident Detection and Response

As highlighted in the analysis, many incidents were not detected quickly because of missing alerts. On top of that, delayed incident reporting and slow debugging process also increase incident resolution time. To address these issues, PT INUSA must improve its monitoring and incident handling strategy. To enhance incident detection, PT INUSA should be more proactive instead of only relying on post-mortem reviews to identify missing alerts. A company wide alert audit should be conducted to standardize monitoring for all services and products to ensure complete alert coverage before incidents occur.

1. Standardizing Alerts Across All Services

   A key improvement is to define a minimum set of standard alerts that every service or product must have. Based on PT INUSA's incident detection analysis, several incidents were prolonged due to missing alerts on key metrics such as transaction success rate drops, CPU utilization spikes, and third-party failures. To prevent these gaps, PT INUSA should enforce a baseline set of mandatory alerts across all systems. Table IV.15 shows the baseline alert that each service must have.

Table IV.15 Baseline Alert Standards for Incident Detection

Category	Required Alert Type	Reason/Justification
Business Impact Metrics	Drop in Successful Transactions per Product Type, Payment Method, and Partner	Many incidents were detected late because no alert was in place to track sudden drops in success rates. This should be standardized across all financial transactions.
System Performance	High CPU Utilization Alert, High Latency, High Error Rate	A previous incident revealed that CPU usage spiked but was not detected due to missing alerts, delaying response times.
External Dependencies	Partner API Response Time & Error Rate	Third-party partner failures are a recurring issue. Automatic monitoring for slow response times or increased failure rates must be implemented across all partner integrations.
Infrastructure Health	Database Query Execution Time Spikes, Database High CPU Utilization, Message Queue High Consumer Lag	Multiple incidents were caused by slow queries, DB locks, or replication lag, which should trigger real-time alerts.

Each team should configure additional alerts tailored to their system's critical flows beyond this baseline.

2. Conducting a Systematic Alert Coverage Audit

   Instead of relying on manual detection of missing alerts during post-mortem reviews, PT INUSA should implement a quarterly or bi-annual alert audit process. This process should:

   - Automatically scan all alert configurations across services.
   - Compare existing alerts against the standardized baseline to identify missing or misconfigured alerts.
   - Implement missing alerts to ensure that critical monitoring is always in place.

Effective incident management does not stop at detection. The incident response process is also critical to minimize business impact. There are several issues identified in incident response at PT INUSA. The main issue was delayed incident reporting. This is the condition where engineers did not report the incident immediately after knowing an issue happen in production environment. Instead, they tried to solve the problem by themselves or just having informal discussions. To address this, PT INUSA should establish a clear on-call SOP that defines engineers' responsibilities when handling incidents. The SOP should require engineers to log incidents immediately after it is detected rather than after deep investigation. This is to ensure that relevant stakeholders are notified about ongoing issue and incident manager can also help to escalate the issue whenever needed. To ensure that engineers follow the SOP, incident reporting should be included as one of on-call performance metrics. The SOP should define the Service-Level Objective (SLO) for incident acknowledgment, for instance, on-call engineers must report an incident within 5–15 minutes after detection. This KPI can be tracked during post-mortem reviews to reinforce accountability. By making this a formal process and doing regular review, PT INUSA can reduce the time between detection and escalation so it will improve the MTTR.

Another critical issue contributing to delayed resolution was gaps in engineers' troubleshooting skills and domain knowledge. In multiple incidents, engineers required prolonged investigation periods before identifying the root cause or temporary resolution, indicating a need for structured competency development. To mitigate this, PT INUSA should introduce regular technical drills or troubleshooting simulations for on-call engineers. Netflix, for example, regularly run failure drills where faults are intentionally injected to their production system (Alvaro, et al., 2016). They found that doing proactive failure testing is a great way for ensuring a reliable product for their members by helping prepare their systems and teams for potential issues in the production environment (Andrus & Schmaus, 2016). Additionally, a structured knowledge-sharing system should be implemented, where engineers document debugging strategies and past resolutions in a centralized repository. This aligns with SRE best practices, where incident retrospectives are leveraged as learning tools to continuously upskill engineering teams.

Another recurring issue was the lack of post-fix monitoring, as demonstrated in several incidents where fixes were deployed but the same issue reoccurred later due to insufficient validation. To prevent this, PT INUSA should enforce a mandatory post-fix monitoring period (e.g., 24-48 hours) where engineers actively track logs and metrics. Additionally, automated validation tests should run post-release to detect anomalies early, and any post-fix anomalies should escalate automatically within the on-call system rather than relying on engineers manually re-detecting the issue.

By implementing all those proposed solutions, PT INUSA is expected to have a better Mean Time to Detect (MTTD) and Mean Time to Resolve (MTTR). This will eventually minimize business impact.

IV.4 Implementation Plan & Justification

This section will present the project timeline to implement the proposed solutions at PT INUSA. As explain in the action priority matrix presented in Figure IV.3, all solutions cannot be implemented immediately. Some solutions, such as alert standardization and change review codeowner validation, can be implemented earlier. Later in a longer term, the progressive rollout strategy and incident knowledge hub will follow because they require significant resources, and they are not very urgent as well. The implementation plan is divided into 5 groups, Detection Improvements, Response Enhancements, Testing & Deployment Improvements, Process Standardization & Automation, and Knowledge Management & Communication. These will be explained in the following sections.

IV.4.1 Implementation Roadmap

To implement the proposed solutions effectively, there should be a structured roadmap that describe the timeline for each initiative, including the owner of the initiative and its milestone. Table IV.16 presents the detailed implementation roadmap. There are five important columns there. The first one is Initiative Name, which is self-explanatory. The second one is Owner, which is the role that is responsible to drive the project. The third one is Months, describing how long the initiative will be done. The fourth one is Key milestones, explaining about the outcome of the initiative. The last one is Frequency, highlighting initiatives that need to be conducted in a regular basis.

The successful execution of these projects requires a strong alignment between many roles in engineering division, such as engineering managers, incident managers, site reliability engineers, devops engineers, and product engineers. This also requires support from engineering leadership. A regular progress checking should be conducted to ensure that the project is still on track or to make immediate action or change of plan if something needs to be changed. If all these projects are implemented successfully, PT INUSA should see a reduction in the number of major incidents as well as a minimized business impact from them.

IV.4.2 Implementation Justification

This section explains the justification for each initiative listed in Table IV.16. Each initiative group will be explained one by one, elaborating about its benefits and aligment with PT INUSA’s goals to reduce and minimize incident impact.

1. Detection Improvements

   Based on analysis result, PT INUSA’s MTTD for major incidents is significantly higher than industry benchmark. To mitigate this, alert standard should be introduced as a baseline alert configuration for each microservices and products. Table IV.15 contains a proposed baseline alert based on the analysis. After the standard is established, alert audit should be conducted to ensure that all services have the mandatory alerts configured. The gaps found from the audit activity then should be implemented. This initiative becomes the highest priority to be done first.

2. Response Enhancements

   Two significant weaknesses were found in analysis, namely delayed incident reporting and slow debugging process. As explained in the previous section, SOP should be made to ensure engineers immediately report incident after detected. The effort to make the SOP is not very high, hence this initiative can be done earlier. To address the slow debugging process, failure simulation exercises should be conducted to improve engineers’ competency in handling incident. However, having this simulation is not easy. The site reliability and infrastructure engineers might need to prepare the infrastructure foundation to make sure that the failure exercise is still controllable and not causing a critical issue on production. Therefore, this initiative is placed a bit later.

3. Testing & Deployment Improvements

   Testing, change review, and deployment issues are the most significant contributors that caused incidents. There are many proposed initiatives under this group. Because of the resource constraint, the first initative to be developed is the implementation of CODEOWER feature to and other merge requests automation checking. After working on this, the DevOps team can then develop the automated deployment log generator and automated configuration checker in the deployment pipeline to reduce human error when doing deployment. Lastly, the biggest initiative is to integrate automation tests to the deployment pipeline. This requires significant effort from engineering teams, especially to develop the gaps in automated tests. Once all done, every new deployment will become much safer because it will always run all critical test cases before the changes go live.

4. Process Standardization & Automation

   In this initiative group, SOPs for deployment and manual change execution can be done first. It will be driven by engineering leader to make sure all engineers execute the deployment or manual change correctly. A bigger initiative, to introduce progressive rollout mechanism, need to be done later because the resource (DevOps team) will focus more on the previous initiatives first that have higher urgency. This mechanism will help deployer to monitor the execution result with minimized risks.

5. Knowledge Management & Communication

   Knowledge sharing session is a relatively easy to implement so it will be done earlier. This should be conducted in a regular basis, e.g., monthly. The standardized post-mortem tagging will support the biggest initiative, incident knowledge hub and analytic dashboard. This platform is designed to help engineers easily learn about past incidents and easily browse or find reference to debug an incident based on previous data. This is a long-term investment to improve incident management practice at PT INUSA.

These solutions are designed to be pragmatic, cost-effective, and aligned with industry best practices. These solutions have been reviewed by a senior engineering leader at PT INUSA. Based on the discussion, the overall strategy was considered feasible and aligned with current organizational priorities. The leader acknowledged that several proposed improvements, such as alert standardization and automation test improvement are already in line with ongoing internal initiatives. Minor additions were suggested to improve testing by also assigning a dedicated test engineer to a major project to help define the test cases. By implementing these initiatives, PT INUSA is expected to see a significant improvement in their incident management key metrics.

Table IV.16 Implementation Roadmap

No	Initiative Name	Owner	Months											Key Milestones	Frequency
			1	2	3	4	5	6	7	8	9	10	11		
Detection Improvements															
1	Define mandatory alert list	Site Reliability Team	🟧											List of mandatory alerts finalized	-
2	Audit existing alerts	Engineering Teams		🟧										Alert gaps are identified	Every 3 months
3	Implement the gap (standardized alerts) based on audit findings	Engineering Teams		🟧	🟧	🟧								Standardized detection rules deployed	
Response Enhancements															
4	Define & enforce on-call SOP (including escalation & resolution timelines)	Incident Management & Site Reliability Teams		🟩										On-call SOP published & enforced	-
5	Conduct failure simulation exercises (chaos testing)	Site Reliability & Engineering Teams					🟩	🟩						First drill completed	Every 6 months
Testing & Deployment Improvements															
7	Integrate automated test execution into the deployment pipeline	QA / Engineering Teams				🟨	🟨	🟨						Tests blocking failed deployments	-
8	Audit automated test coverage & add missing regression test cases	QA / Engineering Teams					🟨	🟨	🟨					Critical flow test coverage improved	Every 6 months
9	Implement CODEOWNER rules, automated MR validation, & linter checks	DevOps & Engineering Teams	🟨	🟨	🟨									Enforced rules in merge process	-

No	Initiative Name	Owner	Months											Key Milestones	Frequency
			1	2	3	4	5	6	7	8	9	10	11		
10	Automate config consistency checks in deployment pipeline	DevOps			🟨									Config validation enabled in CI/CD	-
11	Develop an automated deployment log generator (list all changes & mention change authors)	DevOps & Engineering Teams		🟨	🟨	🟨								First logs automatically generated	-
Process Standardization & Automation															
12	Establish Deployment SOP (announcement, approval from change authors & managers)	Engineering Leadership				🟦								SOP published & enforced	-
13	Establish SOP for manual change execution (post-release monitoring, sanity testing)	Engineering Leadership				🟦								SOP integrated into operations	-
14	Implement progressive rollout (canary deployments) in new infrastructure	DevOps					🟦	🟦	🟦					Progressive rollout available in CI/CD	-
Knowledge Management & Communication															
15	Standardize tagging in post-mortems	Incident Management & Site Reliability Teams				▨								Tagging enforced in all post-mortems	-
16	Develop incident knowledge hub & analytics dashboard	Engineering Leadership & Site Reliability Team							▨	▨	▨	▨	▨	Dashboard providing insights	-
17	Regular incident & debugging knowledge sharing	Engineering Leadership				▨								First session conducted	Every 1 month

Chapter V Conclusion and Recommendation

This chapter summarizes the key findings from the analysis of IT incidents at PT INUSA and provides strategic recommendations for improvement. The conclusions are based on root cause analysis and incident handling evaluation from Chapter IV. The conclusion section below will be presented in the form of question and answer to make sure that all research questions in Chapter I are properly answered. After that, a recommendation section will explore potential future research, and lastly, the contribution section will explain the originality of this research and the contribution to the field of IT incident management in general.

V.1 Conclusion

The findings of this study provide a comprehensive understanding of the primary causes of IT incidents at PT INUSA, as well as the problems in incident detection and response. Based on the analysis of 26 post-mortems, systemic issues are identified and the strategy to address each issue is provided. This research successfully answers the research questions presented in Chapter I:

1. What are the primary causes of IT incidents, particularly in the areas of code, configuration, processes, and testing?

   The analysis revealed that 80% of major incidents were triggered by changes coming from PT INUSA's internal teams, rather than external factors such as traffic spikes or infrastructure issues. This is further supported by the root cause analysis result, which reveals recurring deficiencies in testing and deployment processes. Across all root cause categories, the findings indicate several systemic issues that contributed to multiple incidents:

   - Testing deficiencies

     Many incidents were caused by inadequate staging tests and lack of automated test coverage for critical business flows, allowing undetected issues to reach production.

   - Deployment and change review deficiencies

     Incidents frequently caused by insufficient approval from relevant teams, manually generated deployment logs leading to human error in listing changes, and misconfigured production settings (e.g., missing configurations, incorrect staging values in production) that were not automatically validated before deployment.

To provide a more detailed conclusion, the following breakdown summarizes root causes specific to each category:

- Code-Related Root Causes
    - Logic implementation issues

      Several incidents came from misunderstood business logic, invalid query logic, and unhandled error cases. For example, misinterpretation of timeout handling logic caused incorrect transaction retries.
    - Lack of code review safeguards

      Some incidents occurred because important changes in critical files (such as build configurations) were merged without approval from the right teams.
    - Inefficient Data Handling and Caching Strategies

      Several incidents were caused by suboptimal database queries and caching mechanisms. For example, one incident resulted from the use of local cache (worsen by its misconfiguration) instead of a centralized cache (e.g., Redis), leading to excessive requests to the authentication server and causing a bottleneck. Another issue was slow database queries due to inefficient query logic, which significantly impacted transaction processing times. Additionally, a missing database index in one incident led to high query latency, further degrading system performance.
- Configuration-Related Root Causes
    - Missing or incorrect configuration values

      Several incidents were caused by misconfigured or missing production settings, such as missing production environment variables, incomplete Redis key migrations or incorrect routing configurations during cloud migration.

  - Ineffective configuration validation

    No automated checks existed to ensure production configurations were complete and correct before deployment, allowing staging/test values to accidentally be deployed in production.
  - The use of default settings or copy-paste other service setting

    Several incidents were caused by not modifying default settings. For example, a default and a particular service rate limit setting was used for another service which was expected to have much higher traffic. This caused a lot of requests unserved due to a very low threshold.
  - Manual Certificate Renewal

    Reliance on manual processes during SSL certificate renewal caused delays and disruptions.
- Process-Related Root Causes
  - Deployment execution deficiencies

    Several incidents resulted from accidental deployments as engineers relied on outdated deployment logs that failed to capture all changes being released.
  - Unclear change execution responsibilities

    Some incidents occurred due to confusion over team ownership, particularly for external partner integrations. An example is one of PT INUSA's partner system change their IP address and the notification was not sent to the right team.
- Testing-Related Root Causes
  - Gaps in staging and regression testing

    Many incidents could have been prevented if staging environments better simulated production conditions, and if automated regression tests covered more critical business flows.
  - Lack of automated test enforcement

    Changes were frequently merged without automated test validation and production deployments could proceed even when automated test cases were failed.

2. What factors contribute to delays in incident detection and response, and how do they impact system disruptions?

   The analysis found significant weaknesses in incident detection and response that lead to prolonged downtime and business impact:

   - Incident Detection Issues
     - Eighteen incidents (69%) lacked proper alerts, resulting in delayed detection. Many incidents were first reported by users or internal teams rather than automated monitoring.
     - Missing alerts were most commonly related to transaction drops on specific product, payment method, or partner (supplier), as well as CPU utilization spikes.
     - Reliance on manual monitoring meant that some anomalies were only detected during periodic business reviews, further delaying response time.
   - Incident Response Delays
     - Delayed escalation and informal communication channels led to prolonged resolution times. Some incidents were discussed privately before being officially reported, delaying structured investigation.
     - Lack of structured on-call SOPs makes response processes inconsistent. Engineers were sometimes unsure of their immediate responsibilities.
     - Insufficient post-fix monitoring resulted in issues resurfacing after an initial fix was applied, as teams failed to validate whether their fixes were effective in production.

   These detection and response gaps directly contributed to PT INUSA’s poor Mean Time to Detect (MTTD) and Mean Time to Resolve (MTTR), prolonging the impact of incidents. Strengthening alerting mechanisms, enforcing structured incident response workflows, and improving post-fix monitoring are critical to improving overall resilience.

3. What strategies can be implemented to improve IT incident prevention, detection, and response?

   To mitigate the identified root causes and strengthen incident prevention, detection, and response, several strategic improvements are necessary.

   a. Strengthening Testing and Validation
      - Introduce mandatory regression test audits to assess whether existing automated test suites cover critical business flows. This initiative should includes implementing missing test scenarios.
      - Enforce critical business flows to be automatically tested in every deployment by integrating the automation test to the deployment pipeline and block the deployment when it fails.
      - Adopt progressive rollout strategies that allow changes to be gradually deployed and monitored before full release.

   b. Improving Deployment and Change Review Processes
      - Automate production configuration validation by comparing staging vs. production settings before deployment, preventing incidents caused by missing configurations.
      - Implement CODEOWNER policies and linter rules to enforce strict code and configuration reviews before merging changes. This will automatically prevent changes with insufficient review or approval to be merged.
      - Automate deployment log generation that compare changes with the live version on production. This will ensure the list of changes that will be reviewed is accurate, eliminating human error that can lead to releasing unlisted changes.
      - Establish deployment and manual change execution SOPs. Based on incident analysis, these key activities should be included in the SOP:
        - Require change authors (engineers) to approve or acknowledge their changes in the deployment log. This will ensure engineers are aware that their changes are being

deployed and making sure that all prerequisites for each change have been implemented.

    - Require deployer (change executor) and change authors to actively observe changes in production immediately after deployment or execution. Sanity testing is highly encouraged whenever possible.

c. Enhancing Incident Detection Mechanisms

- Standardize critical alert coverage for all products and services, ensuring that every service has key monitoring rules in place (e.g., success rate drops, CPU spikes, and API failures).
- Conduct periodic alert audits to review existing coverage and implement missing alerts.

d. Streamlining Incident Response and Resolution

- Enforce a structured on-call SOP with response timelines and escalation paths. It should specify how long an on-call engineer must acknowledge, investigate, and escalate an incident. If an issue cannot be resolved within the defined timeline, there must be a clear process for escalating it to senior engineers or managers. This will prevent delays caused by informal coordination and ensure incidents are handled with urgency.
- Require engineers to report anomalies immediately to the incident support channel rather than informally discussing them in private chats.
- Conduct regular failure simulation drills, preparing teams for real-world outages and ensuring faster resolutions.
- Improve post-fix monitoring, requiring engineers to validate and track fixes after deployment to prevent recurrence.

e. Strengthening Knowledge Management and Communication

- Standardize tagging in post-mortems to categorize incidents more effectively, enabling trend analysis and better identification of recurring issues.

- Develop an incident knowledge hub & analytics dashboard that consolidates key insights, root causes, and lessons learned from past incidents, allowing engineers to quickly access relevant case studies.
- Establish regular incident & debugging knowledge-sharing sessions to ensure that engineers across teams can learn from past incidents, improve troubleshooting skills, and apply best practices proactively.

V.2 Recommendation

The analysis in this study has identified systemic gaps in incident prevention, detection, and response at PT INUSA. The proposed strategic improvements serve as a roadmap for enhancing operational resilience. Based on the implementation roadmap, PT INUSA should prioritize the following areas to achieve tangible improvements in incident management:

- Testing & Deployment Enhancements

  Strengthening pre-deployment validation (automated tests, progressive rollout, and enforced change approvals) to reduce deployment-related incidents.
- Incident Detection Maturity

  Standardizing critical alert coverage and conducting periodic audits to ensure monitoring gaps are closed.
- Response Readiness & Accountability

  Implementing clearer on-call SOPs, escalation paths, and structured post-fix monitoring to ensure incidents are resolved more efficiently.
- Knowledge Sharing & Post-Mortem Utilization

  Creating an incident knowledge hub and regular incident-sharing sessions to prevent repeated mistakes and foster cross-team learning.

These priorities should be executed gradually due to resource limitation. This is important to ensure that immediate risks can be addressed first while long-term improvements are gradually implemented.

Even though this study provides a comprehensive assessment of PT INUSA’s incident management gaps and solutions, future research could further refine these findings by exploring the integration of Artificial Intelligence for IT Operations (AIOps) to PT INUSA's incident management processes. AIOps is the application of AI and machine learning (ML) in IT operations to enable intelligent automation, predictive analytics, and real-time anomaly detection to improve system reliability. One of industry examples is Netflix. They have developed an auto-remediation system that combines rule-based classifiers with machine learning to automatically fix failed data platform jobs. This has successfully solved a significant percentage of memory configuration errors, reducing manual operational effort (Hou, et al., 2024). Another example is Google. In managing its cloud infrastructure, Google applies AI and machine learning to add intelligence to its operations. They implemented it in their cloud networking platform, showing how AIOps can proactively optimize IT operations (Google Research, 2022).

By adopting AIOps, PT INUSA can shift from a reactive to a proactive incident management approach. Future research can explore the development of custom AIOps models that are designed specificly for PT INUSA’s infrastructure and incident patterns. It utilizes real-time data to improve automation and continuous improvement.

V.3 Contribution

This research provides both scientific and practical contributions to the field of IT incident management, particularly in understanding the root causes of IT incidents and improving incident detection and response. It offers an in-depth analysis of systemic failure patterns within a real-world fintech environment. While prior studies have largely focused on incident response strategies or broader industry-wide and IT infrastructure outage patterns, this research takes a company-specific perspective, focusing on application-layer incidents. It examines how internal factors—such as code defects, misconfigurations, process gaps, and testing deficiencies—contribute to failures. A key finding of this study is that insufficient change reviews, particularly the lack of approval from code owners, is one of the primary causes of incidents. This specific relationship between insufficient code review and incident occurrence has not been explicitly discussed in previous research.

From a practical standpoint, this study provides data-driven recommendations to improve incident prevention, detection, and response. It identifies critical gaps in alerting mechanisms, incident escalation, and post-fix monitoring, all of which have contributed to prolonged downtime. As a direct outcome, the study proposes a baseline alert standard to enhance early detection of transaction failures, CPU spikes, and third-party integration issues. This proposed standard is applicable to similar fintech organizations seeking to improve their monitoring frameworks. Additionally, structured improvements—such as progressive deployment rollouts, automated deployment change logs, automated configuration checks, end-to-end testing in deployment pipelines, and failure simulation exercises—provide practical strategies to strengthen incident management. By addressing both technical gaps and process inefficiencies, this research presents a comprehensive framework for reducing incident frequency and enhancing response effectiveness in fintech and other technology-driven companies.

REFERENCES

Aceto, G., Botta, A., Marchetta, P., Persico, V., & Pescape, A. (2018). A comprehensive survey on internet outages. *Journal of Network and Computer Applications*, 36-63.

Alt, R., Leimeister, J. M., Priemuth, T., Sachse, S., Urbach, N., & Wunderlich, N. (2020). Software-Defined Business. *Business & Information Systems Engineering*, 609-621.

Alvaro, P., Andrus, K., Sanden, C., Rosenthal, C., Basiri, A., & Hochstein, L. (2016). Automating Failure Testing Research at Internet Scale. *Seventh ACM Symposium on Cloud Computing (SoCC 2016)* (pp. 181-193). Santa Clara: ACM (Association for Computing Machinery).

Andrus, K., & Schmaus, B. (2016, January 20). *Automated Failure Testing.* Retrieved from Netflix Technology Blog: http://techblog.netflix.com/2016/01/automated-failure-testing.html

Atlassian. (2024). *Escalation policies for effective incident management.* Retrieved from Atlassian: https://www.atlassian.com/incident-management/on-call/escalation-policies

AXELOS. (2019). *ITIL Foundation (ITIL 4 Foundation).* Norwich: TSO (The Stationery Office).

Bashir, A., & Soomro, T. R. (2012). Comparative Study on Incident Management. *International Journal of Applied Information Systems (IJAIS)*.

Batta, R., Shwartz, L., Nidd, M., Azad, A. P., & Kumar, H. (2021). A system for proactive risk assessment of application changes in cloud operations. *IEEE 14th International Conference on Cloud Computing (CLOUD).* Chicago: IEEEE.

Beyer, B., Jones, C., Petoff, J., & Murphy, N. R. (2016). *Site Reliability Engineering: How Google Runs Production Systems.* Sebastopol: O'Reilly Media.

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology*, 77-101.

Chen, Z., Kang, Y., Li, L., Zhang, X., Zhang, H., Xu, H., . . . Lyu, M. R. (2020). Towards intelligent incident management: why we need it and how we make it. *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (pp. 1487-1497). Association for Computing Machinery.

Cichonski, P., Millar, T., Grance, T., & Scarfone, K. (2012). *Computer Security Incident Handling Guide.* Gaithersburg: National Institute of Standards and Technology.

Elliott, V. (2018). Thinking about the Coding Process in Qualitative Data Analysis. *The Qualitative Report*, 2850-2861.

Google Research. (2022). *AIOps in Google Cloud.* Retrieved from Google Research: https://research.google/pubs/aiops-in-google-cloud/

Gunawi, H. S., Hao, M., Suminto, R. O., Laksono, A., Satria, A. D., Adityatama, J., & Eliazar, K. J. (2016). Why Does the Cloud Stop Computing?: Lessons from Hundreds of Service Outages. *SoCC '16: Proceedings of the Seventh ACM Symposium on Cloud Computing* (pp. 1-16). New York: Association for Computing Machinery.

Güven, S., & Murthy, K. (2016). Understanding the Role of Change in Incident Prevention. *2016 12th International Conference on Network and Service Management (CNSM)* (pp. 268-271). Montreal: IEEE.

Guy Carpenter. (2024, August 1). *A Closer Look: Unveiling The Global Impact of Crowdstrike Event.* Retrieved from GuyCarpenter: https://www.guycarp.com/content/dam/guycarp-rebrand/insights-images/2024/07/2024-8-1-Unveiling-the-Global-Impact-of-CrowdStrike-Event.pdf

Hou, B., Tamayo, S. V., Chen, X., Tian, L., Ristow, T., Wang, H., . . . Dixit, P. (2024, March 5). *Evolving from Rule-based Classifier: Machine Learning Powered Auto Remediation in Netflix Data Platform.* Retrieved from Netflix Technology Blog: https://netflixtechblog.com/evolving-from-rule-based-classifier-machine-learning-powered-auto-remediation-in-netflix-data-039d5efd115b

International Organization for Standardization. (2018). *Information technology — Service management — Part 1: Service management system requirements (ISO/IEC Standard No. 20000-1:2018).* Geneva: International Organization for Standardization.

ITIC. (2017, June 22). *Average cost per hour of server downtime worldwide in 2017, by vertical industry (in million U.S. dollars) [Graph].* Retrieved from Statista: https://www.statista.com/statistics/780699/worldwide-server-hourly-downtime-cost-vertical-industry/

Juran, J. M., & Feo, J. A. (2010). *Juran's Quality Handbook: The Complete Guide to Performance Excellence, Sixth Edition.* New York: McGraw-Hill.

Kapel, E. (2023). Incident Prevention Through Reliable Changes Deployment. *2023 IEEE/ACM 45th International Conference on Software Engineering:*

Companion Proceedings (ICSE-Companion) (pp. 200-202). Melbourne: IEEE.

Katragadda, S. R., Peddinti, S. R., Pandey, B. K., & Tanikonda, A. (2021). Machine Learning-Enhanced Root Cause Analysis for Accelerated Incident Resolution in Complex Systems. *Journal of Science & Technology*.

Kiger, M. E., & Varpio, L. (2020). Thematic analysis of qualitative data: AMEE Guide No. 131. *Medical Teacher*, 846-854.

Kosińska, J., Baliś, B., Konieczny, M., Malawski, M., & Zieliński, S. (2023). Toward the Observability of Cloud-Native Applications: The Overview of the State-of-the-Art. *IEEE Access*, 73036-73052.

Kumar, S., Li, A., Wong, H., Chauhan, H., Shubhankar, S., & Oetama, I. (2023, March 29). *Indonesia's Fintech Industry Is Ready to Rise.* Retrieved from BCG: https://www.bcg.com/publications/2023/fintech-industry-indonesia-growth

Lawless, J. (2023, November 3). *ITIL Deployment Management.* Retrieved from Purple Griffon: The IT Service Management Training & Consultancy Specialists: https://purplegriffon.com/blog/itil-deployment-management-practice

Lawless, J. (2023, November 9). *ITIL Incident Management - Everything You Need To Know.* Retrieved from Purple Griffon: https://purplegriffon.com/blog/itil-incident-manager

Lawless, J. (2024, February 29). *ITIL Service Validation And Testing.* Retrieved from Purple Griffon: The IT Service Management Training & Consultancy Specialists: https://purplegriffon.com/blog/itil-service-validation-and-testing

Lawrence, A., & Simon, L. (2023). *Annual Outages Analysis 2023.* New York: Uptime Institute.

Lee, J. (2021, October 4). *Here's How Many Millions in Ad Revenue Facebook Could Have Lost During Outage.* Retrieved from Snopes: https://www.snopes.com/news/2021/10/04/facebook-ad-revenue/

Morris, K. (2016). *Infrastructure as Code: Managing Servers in the Cloud.* O'Reilly Media, Inc.

New Relic. (2023). *State of Observability: Service-level metrics.* Retrieved from New Relic: https://newrelic.com/resources/report/observability-forecast/2023/state-of-observability/service-level-metrics

Okes, D. (2009). *Root Cause Analysis: The Core of Problem Solving and Corrective Action.* Milwaukee: ASQ Quality Press.

Patel, M., & Patel, N. (2019). Exploring Research Methodology: Review Article. *International Journal of Research and Review*, 48-55.

Prince, S. (2016, February 11). *The Product Managers' Guide to Continuous Delivery and DevOps.* Retrieved from Mind the Product: https://mtp2017.wpenginepowered.com/what-the-hell-are-ci-cd-and-devops-a-cheatsheet-for-the-rest-of-us/

Reid, I., & Smyth-Renshaw, J. (2012). Exploring the Fundamentals of Root Cause Analysis: Are We Asking the Right Questions in Defining the Problem? *Quality and Reliability Engineering International*, 535-545.

Richard, Gaol, F. L., Warnars, H. L., Abdurachman, E., & Soewito, B. (2019). Development of Web Application based on ITIL – Incident Management Framework In Computer Laboratory. *2019 International Conference on Information Management and Technology (ICIMTech)* (pp. 120-125). Jakarta/Bali: IEEE.

Rocco, T. S., & Plakhotnik, M. S. (2009). Literature Reviews, Conceptual Frameworks, and Theoretical Frameworks: Terms, Functions, and Distinctions. *Human Resource Development Review*, 120-130.

Saha, A., & Hoi, S. C. (2022). Mining root cause knowledge from cloud service incident investigations for AIOps. *ICSE-SEIP '22: Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice* (pp. 197-206). New York: Association for Computing Machinery.

Shahin, M., Babar, M. A., & Zhu, L. (2017). Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices. *IEEE Access*, 3909-3943.

APPENDICES

Appendix A Incident Root Cause Categories at PT Inovindo Nusakarya

Root Cause Category	Description
Code	Refers to bugs in the application code or limitations in the code implementation that prevent certain functions from running as expected. Some features may still operate with restrictions.
Configuration	Occurs when there is an error, omission, or inadequacy in the configuration process, which may cause some or all functions to fail.
Cyber Security Attack	Refers to an intentional, malicious attempt to compromise the confidentiality, integrity, or availability of IT systems. This includes attacks such as malware, phishing, ransomware, denial-of-service (DoS), and other security threats.
Human Error	Occurs when mistakes are made by people, often due to lack of knowledge, experience, or attention.
Infrasructure	Relates to failures or issues within network or infrastructure elements (e.g., database, firewall, load balancer).
Knowledge	Refers to a lack of necessary information during system development, resulting in limited functionality or restricted system capabilities.
Not an Incident	Describes an event that initially appears to be an incident but is later determined not to be, due to misclassification or misunderstanding.
Not Technology Incident	Refers to events or issues that do not involve a failure or disruption in technology systems, often arising from non-technical causes or misunderstandings that do not require technical intervention.
Process and Policy	Refers to violations or misapplications of internal processes and policies (e.g., failure to follow a review process or incomplete requirements) that lead to errors or inconsistencies in functionality. It also includes cases where no process or policy exists to cover a specific scenario, leading to incidents.
Technology – Design Architecture	Refers to incidents caused by the complexity or limitations of the system's architecture. While the design may have originally met business requirements, over time, as the platform grows and evolves, the architecture can become a source of incidents due to its inherent complexity or outdated design choices based on earlier requirements
Testing	Refers to incomplete or inadequate testing processes (e.g., missing test scenarios or unclear expectations) that result in errors during deployment or operation.
Third Party	Refers to problems that arise solely due to third-party providers or partners.
Unknown	Refers to root causes that are not reproducible or identifiable by engineers, and remain unresolved due to lack of evidence.

Appendix B List of Tags and Its Description

Tag	Description
Accidental Deployment	A change or update that was unintentionally deployed to production due to miscommunication, procedural gaps, or human error.
Change Approval Bottleneck	Delays in deployment due to slow approval processes, often caused by unclear responsibilities, lack of automation, or excessive manual steps.
Change Execution Process Deficiency	Weaknesses in the process of implementing changes, such as insufficient validation (lack of monitoring activity) and lack of rollback plans.
Configuration Error	Incorrect settings in configuration files that cause system failures, misbehavior, or degraded performance.
Configuration Management Deficiency	Lack of proper version control, auditing, or enforcement of configuration standards, leading to inconsistencies between environments.
Delayed Resolution	An incident that remained unresolved for an extended period due to slow investigation, ineffective escalation, or resource unavailability.
Dependency Between Base App and Feature Module	Tight coupling between core application components and additional feature modules, leading to failures when dependencies are not properly managed.
Deployment Process Deficiency	Weaknesses in the deployment process, such as lack of validation steps and lack of automation to prevent human error throughout the deployment process.
Idempotency Check Issue	Failure to properly implement idempotency checks, resulting in duplicate transactions or repeated execution of actions that should only run once.
Improper Load Management	Inability to handle spikes in traffic due to lack of scaling mechanisms, inefficient resource allocation, or absence of rate-limiting strategies.
Incident Triggered by Change	An incident that occurred as a direct result of a recent code, configuration, or infrastructure change
Incomplete Monitoring	Monitoring gaps where critical system behaviors or metrics were not tracked, making it difficult to detect and diagnose issues.
Incomplete Regression Test Coverage	Lack of sufficient regression tests to catch unintended side effects of new code changes before deployment.
Incomplete Test Coverage	Absence of test cases for certain scenarios, leading to undetected defects in production.
Incomplete Validation Logic	Business rules or logic checks that are not fully implemented, allowing incorrect or unexpected behavior to occur.
Increased Traffic	A sudden increase in user activity that overwhelms system resources, leading to degraded performance or failures
Informal Communication Channels	Use of personal chats, verbal discussions, or unofficial messages for critical incident handling, leading to delays and misalignment.

Tag	Description
Infrequent Deployment	Long intervals between deployments, increasing the risk of releasing large changes that are harder to test and rollback or reducing the familiarity with the service.
Insufficient Change Review Process	Lack of thorough peer review or expert validation before merging and deploying changes.
Invalid Query Logic	Flaws in database queries, such as incorrect joins, missing filters, or suboptimal indexing, leading to incorrect or slow data retrieval.
Knowledge Transfer and Handover Deficiency	Poor documentation and lack of structured handover processes, resulting in knowledge gaps when team members transition.
Lack of Knowledge	Engineers or teams not having sufficient understanding of the systems they manage, leading to misconfigurations, wrong code implementation, improper fixes, or slow troubleshooting.
Lack of Linter or Code Checker	Absence of automated tools to enforce coding standards, detect syntax errors, and prevent stylistic inconsistencies in the codebase.
Lack of Migration Strategy Guideline	No standardized approach for migrating systems, configurations, or data, leading to overlooked dependencies and missing components.
Lack of Monitoring Activity	Engineers failing to actively monitor system behavior after a deployment or configuration change, leading to delayed incident detection.
Lack of Staging Tests	Critical features or workflows not being tested in a staging environment before being released to production.
Limited Pre-deployment Check	Inadequate validation before deployment, such as skipping checklist reviews, failing to verify expected outcomes, or missing approvals
Local cache usage	Storing temporary data in a local cache instead of a centralized cache, leading to inconsistencies across distributed systems.
Logic Misunderstanding	Developers misinterpreting how existing code works, leading to incorrect modifications that introduce bugs.
Manual Certificate Renewal	SSL/TLS certificates requiring manual renewal, increasing the risk of unexpected service disruptions when they expire.
Manual Rollback Process	Rollback procedures that require manual intervention instead of automated rollback mechanisms, increasing downtime in case of failure.
Manual Scale Server	Server capacity adjustments requiring manual actions rather than automated scaling, leading to slow response during traffic spikes
Manual Switch Partner	Switching third-party service providers (e.g., payment gateways) manually instead of using automated failover mechanisms.
Miscommunication Partner Notification	External partners failing to properly communicate planned changes, leading to misalignment and service disruptions.
Misconfigured Cache Mechanism	Incorrect caching configurations causing stale data, excessive cache invalidation, or unexpected system behavior.

Tag	Description
Missed Code Implementation	A required functionality that was intended but never fully implemented, leading to incomplete features.
Missing Alerts	Critical system events not triggering alerts, causing delayed detection and response to incidents.
Missing Configuration	Configuration settings missing from production environments, causing service failures or degraded performance.
Missing Database Index	Absence of necessary indexes on database tables, leading to inefficient queries and slow performance.
Missing Feature in Migration	A feature or component that was present in the old system but was not migrated properly to the new system.
Missing Requirements	Functional or technical requirements that were either not defined or not communicated properly, resulting in incomplete implementations.
Slave DB Lag	A delay in database replication between primary and secondary (slave) databases, causing inconsistent data reads.
Testing Challenges	Difficulties in executing tests due to infrastructure limitations, lack of proper test data, or unreliable test environments.
Timeout Misinterpretation	Incorrect handling of timeout scenarios, leading to failed transactions or unintended retries that worsen system performance.
Unclear Ownership and Communication	Confusion over which team or individual is responsible for a particular service or incident, causing an incident or leading to delays in resolution.
Unfamiliarity of Service Deployment	Engineers lacking experience or knowledge about the correct procedures for deploying a specific service.
Unhandled/Mishandled Error in Code	Errors in code that are not properly caught or handled, leading to unexpected failures or crashes.
Unimplemented Callback Handling	Failure to implement logic to process asynchronous callbacks from external services, causing incomplete workflows.
Unlisted Change in the Deployment Document	A change that was deployed but was not documented in the deployment log, leading to unexpected modifications in production.
Unoptimized Code	Inefficient code implementation that leads to performance issues, excessive resource consumption, or slow execution times.

Appendix C List of Major Incidents

Incident Number	Title
2024080704	QRIS & topup transaction stuck in pending after deployment of payment service
2024080501	Stuck Transaction Using Payment Method A Due to unhandled asynchronous payment from the 3rd party
2024071101	Push notification doesn't work on app version x.xx
2024070301	Timeout Causing Double Remittance
2024070202	Increasing Stuck Transaction Due to auth server exploded
2024070102	Electricity Prepaid Transactions Failed due to Partner A IP closed
2024052901	QRIS registration screen crash since app version x.xx
2024042201	Some users are getting stuck on a page in the app when trying to open the QRIS feature
2024042101	Transactions got refunded although transaction status in Partner B actually succeeded
2024031903	E-money transaction bad gateway
2024022901	Stuck Transaction in Paid State Increase for Prepaid Phone Credit Products
2024021901	There is no pulsa and paket data transaction to the Partner C
2024020601	Flash sale Got Error Out of Stock
2024013001	Error create transaction on 3PL courier
2024011801	Mix Top Up didn't work for Blacklisted Refund Topup Users
2023112401	Partner D Kirim Uang Failed
2023110301	Invoice Auto Expired for Product A Categories
2023102601	No BPJS Transaction since 25-10-2023 at 16.10 until 26-10-2023 at 10.24
2023101901	Kirim Uang Stuck Transaction Spiking
2023101702	Can't use Payment Method B because of loan wallet duplication
2023101701	Accidentally deployed previously reverted commits in Service A
2023092202	Service B DB CPU Usage 100%
2023092102	Service C Down After Cutover to GCP Shared Kafka
2023091301	Fail to Receive Requests from Partner E
2023081501	Product B cannot pay with Payment Method D
2023081001	Rate Limit affecting Product D transaction

Appendix D Summary of Incident Tags and Their Occurrences

Tag	Number of Occurrences
Incident Triggered by Change	21
Missing Alerts	18
Lack of Staging Tests	9
Delayed Resolution	8
Deployment Process Deficiency	6
Insufficient Change Review Process	5
Missing Configuration	4
Unclear Ownership and Communication	4
Incomplete Test Coverage	4
Invalid Query Logic	3
Incomplete Regression Test Coverage	3
Lack of Migration Strategy Guideline	3
Change Execution Process Deficiency	3
Lack of Monitoring Activity	3
Accidental Deployment	2
Lack of Knowledge	2
Unhandled/Mishandled Error in Code	2
Knowledge Transfer and Handover Deficiency	2
Incomplete Validation Logic	2
Misconfigured Cache Mechanism	1
Testing Challenges	1
Missing Database Index	1
Increased Traffic	1
Configuration Error	1
Informal Communication Channels	1
Incomplete Monitoring	1
Lack of Linter or Code Checker	1
Missing Requirements	1
Improper Load Management	1
Dependency Between Base App and Feature Module	1
Infrequent Deployment	1
Unlisted Change in the Deployment Document	1
Configuration Management Deficiency	1
Missed Code Implementation	1
Limited Pre-deployment Check	1
Change Approval Bottleneck	1
Local cache usage	1

Tag	Number of Occurrences
Missing Feature in Migration	1
Logic Misunderstanding	1
Slave DB Lag	1
Manual Certificate Renewal	1
Timeout Misinterpretation	1
Manual Rollback Process	1
Unfamiliarity of Service Deployment	1
Unoptimized Code	1
Unimplemented Callback Handling	1
Manual Switch Partner	1
Idempotency Check Issue	1
Miscommunication Partner Notification	1
Manual Scale Server	1